CARATULA

Resumen

todo

Agradecimientos

todo

Índice general

1

Probabilidad y Estadística

Las expectativas sobre los objetivos que la inteligencia artificial podrá alcanzar en los próximos años tienden a ser muy ambiciosos. Avances constantes y no circunstanciales, solo serán posibles con el entendimiento absoluto en la materia, alejándose del enfoque de prueba y error.

El boom de la *inteligencia artificial* llegó para quedarse. Cada vez más, las decisiones asociadas a actividades comerciales y/o tecnológicas son tomadas en base a resultados algorítmicos. Dichas decisiones, lejos de basarse en reglas rígidas creadas por programadores, son tomadas en base al reconocimiento de patrones observados en experiencias estadísticas previas, definiendo lo que se conoce como el *aprendizaje estadístico*. Aunque la inteligencia artificial nos ha llevado a conseguir logros notables en las últimas décadas, las expectativas por lo que este campo será capaz de alcanzar en los próximos años tienden a exagerarse más de lo que podría ser posible. Para evitar todo tipo de decepción, es imprescindible entonces lograr avances permanentes que eviten todo tipo de estancamiento. Avances constantes y no circunstanciales, solo serán posibles con el entendimiento absoluto en la materia, alejándose del enfoque de *prueba y error*. El estudio de la matemática detrás de estos métodos es indispensable para transitar este camino. Quizás la única manera de comprender a la máquina sea con matemática.

1.1. Teoría de Probabilidad

La probabilidad es una teoría matemática que estudia el azar y la incertidumbre por medio de experimentos aleatorios [1]. La misma, se desarrolla en un marco conocido como **espacio de probabilidad**. Un espacio de probabilidad $(\Omega, \mathcal{A}, \mathbb{P})$ consta de una terna formada por un **espacio muestral** Ω que posee todos los resultados posibles de un experimento aleatorio, una **sigma-álgebra** $\mathcal{A}$ que contiene todos los conjuntos medibles, y una **medida de probabilidad** $\mathbb{P}$ que justamente mide que tan probable es un evento; cada uno con ciertas características. Las dos fórmulas más características para una $\mathbb{P} : \mathcal{A} \to [0, 1]$ son la **fórmula de probabilidades totales** y la **probabilidad condicional**.

Propiedades 1.1 *1. Sean $A_1, \cdots A_m \in \mathcal{A}$ una partición de Ω (es decir $A_i \cap A_j = \emptyset \quad \forall\, i \neq j$ y $\bigcup_{i=1}^m A_i = \Omega$), entonces para todo $B \in \mathcal{A}$:*

$$\mathbb{P}(B) = \sum_{i=1}^{m} \mathbb{P}(A_i \cap B) \tag{1.1}$$

2. Sea $B \in \mathcal{A}$ un evento con $\mathbb{P}(B) > 0$, entonces

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)} \tag{1.2}$$

Combinando estas dos fórmulas surge la famosa **regla de bayes**:

$$\mathbb{P}(A_k|B) = \frac{\mathbb{P}(B|A_k)\mathbb{P}(A_k)}{\sum_{i=1}^m \mathbb{P}(B|A_i)\mathbb{P}(A_i)} \tag{1.3}$$

Un concepto relacionado con estas expresiones matemáticas es lo que se conoce como **independencia estadística**.

Definición 1.1 *Dos eventos $A, B \in \mathcal{A}$ son independientes si $\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B)$.*

1.1.1. Variables Aleatorias

Distribución	Notación	$P_X(x)$	Soporte	$\mathbb{E}[X]$	$\text{var}(X)$
Bernoulli	$\text{Ber}(p)$	$p^x(1-p)^{1-x}$	$\{0,1\}$	p	$p(1-p)$
Binomial	$\text{Bin}(n,p)$	$\binom{n}{x}p^x(1-p)^{n-x}$	$\{0,\cdots,n\}$	np	$np(1-p)$
Geométrica	$\text{Geo}(p)$	$(1-p)^{x-1}p$	$\mathbb{N}$	$\frac{1}{p}$	$\frac{1-p}{p^2}$
Pascal	$\text{Pas}(k,p)$	$\binom{x-1}{k-1}(1-p)^{x-k}p^k$	$\{k, k+1, \cdots\}$	$\frac{k}{p}$	$k\frac{1-p}{p^2}$
Poisson	$\text{Poi}(\mu)$	$\frac{\mu^x e^{-\mu}}{x!}$	$\mathbb{N}_0$	μ	μ

Cuadro 1.1: Algunas de las variables discretas más habituales.

Trabajar con eventos genéricos puede ser tedioso. Las **variables aleatorias** son funciones $X : \Omega \to \mathbb{R}$ que permiten codificar los resultados de un experimento aleatorio en valores numéricos. La medida de probabilidad sobre una variable aleatoria se caracteriza con la **función de distribución** $F_X(x) = \mathbb{P}(X \leq x)$. Se denomina **átomo** a todo $x \in \mathbb{R}$ tal que $\mathbb{P}(X = x) > 0$ (discontinuidades de la función de distribución). Si la suma de las probabilidades de los átomos vale 1 la variable se denomina discreta. Si en cambio, la función de distribución es continua, la variable se denomina continua. En caso de no ser ni continua ni discreta se denomina variable mixta.

El uso de la función de distribución para efectuar cálculos sigue siendo un poco incómodo, por eso surgen la **función de masa de probabilidad** $P_X(x) = \mathbb{P}(X = x)$ para

variables discretas y la **función de densidad de probabilidad** $p_X(x) = F'_X(x)$ para las variables continuas[1]. Cuando se habla de **distribución** a secas, se hace referencia a la medida de probabilidad asociada, siendo indiferente si la variable se representa con la función de distribución, la masa de probabilidad o la densidad. Cuando se quiere hablar en general, y no se sabe si la variable es discreta o continua, se usará la notación p_X la cuál debe interpretarse como masa o densidad según corresponda. En este sentido, $p_X(x)$ debe ser una función no negativa que integre 1[2]. Se denomina **soporte** a $\mathcal{X} = \{x \in \mathbb{R} : \ p_X(x) > 0\}$[3]. Algunas de las variables más conocidas pueden verse en el Cuadro 1.1 para el caso de las variables aleatorias discretas y en el Cuadro 1.2 para las variables continuas[4].

Distribución	Notación	$p_X(x)$	Soporte	$\mathbb{E}[X]$	$\text{var}(X)$
Uniforme	$\mathcal{U}(a,b)$	$\frac{1}{b-a}$	$[a,b]$	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$
Normal	$\mathcal{N}(\mu,\sigma^2)$	$\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	$\mathbb{R}$	μ	σ^2
Exponencial	$\exp(\lambda)$	$\lambda e^{-\lambda x}$	$[0,\infty)$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$
Gamma	$\Gamma(\nu,\beta)$	$\frac{\beta^\nu}{\Gamma(\nu)}x^{\nu-1}e^{-\beta x}$	$[0,\infty)$	$\frac{\nu}{\beta}$	$\frac{\nu}{\beta^2}$
Beta	$\beta(a,b)$	$\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}x^{a-1}(1-x)^{b-1}$	$[0,1]$	$\frac{a}{a+b}$	$\frac{ab}{(a+b)^2(a+b+1)}$
Chi cuadrado	χ^2_k	$\frac{1}{2^{\frac{k}{2}}\Gamma\left(\frac{k}{2}\right)}x^{\frac{k}{2}-1}e^{-\frac{x}{2}}$	$[0,\infty)$	k	$2k$
t-student	t_ν	$\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{t^2}{\nu}\right)^{-\frac{\nu+1}{2}}$	$\mathbb{R}$	0	$\frac{\nu}{\nu-2}$
Lomax	$\text{Lomax}(\alpha,\beta)$	$\frac{\alpha\beta^\alpha}{(x+\beta)^{\alpha+1}}$	$[0,\infty)$	$\frac{\beta}{\alpha-1}$	$\frac{\beta^2\alpha}{(\alpha-1)^2(\alpha-2)}$

Cuadro 1.2: Algunas de las variables discretas más habituales.

Así como existen las variables aleatorias, se pueden definir los vectores aleatorios. En este contexto, las distribuciones involucradas reciben el nombre de conjunta (por ejemplo $p_{XY}(x,y)$), marginal ($p_X(x)$ y $p_Y(y)$) y condicional ($p_{X|Y=y}(x)$ y $p_{Y|X=x}(y)$). Las Props. 1.1 tendrán su contraparte con las funciones masa o las densidades.

Propiedades 1.2 *1. Las distribuciones marginales pueden calcularse como*

$$p_Y(y) = \int_{\mathcal{X}} p_{XY}(x,y)dx \tag{1.4}$$

[1]Siendo rigurosos, las variables con densidad son un subgrupo dentro de las variables continuas llamadas *absolutamente continuas*.

[2]En el caso de variables discretas, deben intercambiarse las integrales asociadas a dichas variables por sumas o series según corresponda.

[3]Siendo rigurosos, el soporte es la *clausura* de este conjunto.

[4]La varianza de la t-student solo existe para $\nu > 2$. La media de la Lomax solo existe para $\alpha > 1$ y su varianza para $\alpha > 2$.

2. Las distribuciones condicionales pueden calcularse como

$$p_{Y|X=x}(y) = \frac{p_{XY}(x,y)}{p_X(x)} \tag{1.5}$$

De igual manera, dos variables aleatorias (X, Y) son independientes si y solo si su distribución conjunta se puede factorizar como $p_{XY}(x, y) = p_X(x)p_Y(y)$. Algunos de los vectores aleatorios más conocidos pueden verse a continuación.

Definición 1.2 *El vector aleatorio $X = (X_1, \cdots, X_d)$ tiene distribución normal multivariada $X \sim \mathcal{N}(\mu, \Sigma)$ si su densidad conjunta es de la forma:*

$$p(X_1, \cdots, X_d) = \frac{1}{(2\pi)^{\frac{d}{2}}\sqrt{|\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} \tag{1.6}$$

De igual manera, el vector aleatorio $X = (X_1, \cdots, X_d)$ tiene distribución multinomial $X \sim \mathcal{M}_n(n, [p_1, \cdots, p_d])$ si su función de probabilidad conjunta es de la forma:

$$P(X_1, \cdots, X_d) = \frac{n!}{\prod_{i=1}^d x_i!} \left(\prod_{i=1}^d p_i^{x_i}\right) \mathbf{1}\left\{\sum_{i=1}^d x_i = n, x \in \mathbb{N}_0^d\right\} \tag{1.7}$$

La probabilidad conjunta, siguiendo (1.5), siempre se puede factorizar de la forma $p_{XY}(x, y) = p_{Y|X=x}(y)p_X(x)$. Dado que en probabilidad este tipo de descomposiciones es única, cuando la conjunta es factorizada inmediatamente se conoce la marginal y la condicional. El siguiente ejemplo muestra como se puede factorizar analíticamente este tipo de conjunta.

Ejemplo 1.1 *Sea $p_{XY}(x, y) = e^{-x}\mathbf{1}\{0 < y < x\}$, hallar $p_X(x)$ y $p_{Y|X=x}(y)$.*

En la factorización $p_{XY}(x, y) = p_{Y|X=x}(y)p_X(x)$, la variable y aparece solamente en una de las distribuciones, por lo que el primer paso es separar todos los lugares donde aparezca dicha variable $p_{XY}(x, y) = \mathbf{1}\{0 < y < x\} \cdot e^{-x}\mathbf{1}\{x > 0\}$. Luego, completando la densidad condicional para que integren 1 en y (ayudándose con el Cuadro 1.2), puede verse que

$$p_{XY}(x, y) = \underbrace{\frac{1}{x}\mathbf{1}\left\{0 < y < x\right\}}_{p_{Y|X=x}(y)} \cdot \underbrace{xe^{-x}\mathbf{1}\left\{x > 0\right\}}_{p_X(x)} \tag{1.8}$$

En este caso, del Cuadro 1.2, se deduce que $Y|X = x \sim \mathcal{U}(0, x)$ y $X \sim \Gamma(2, 1)$.

1.1.2. Momentos

En muchas circunstancias, tener toda una función que caracterice un experimento aleatorio es abrumador. Se denominan **momentos** a las magnitudes más relevantes que caracterizan a las variables aleatorias, caracterizadas por el operador esperanza $\mathbb{E}[\cdot]$. Los más representativos son la media $\mathbb{E}[X] = \int_{\mathcal{X}} x \cdot p_X(x)dx$, que representa a la variable aleatoria como una constante, y la varianza $\text{var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$, que representa el error que

se comente al aproximar a la variable aleatoria con su media. Algunas de las propiedades básicas de los momentos se presentan a continuación.

Propiedades 1.3 *Propiedades de la esperanza.*

1. *Cuando se aplican funciones sobre variables aleatorias, no es necesario conocer la distribución de la nueva variable* $\mathbb{E}[g(X)] = \int_{\mathcal{X}} g(x)p_X(x)dx$.
2. *La esperanza es lineal* $\mathbb{E}[aX + b] = a \cdot \mathbb{E}[X] + b$.
3. *Las probabilidades son un caso particular de las esperanzas* $\mathbb{E}[\mathbf{1}\{X \in A\}] = \mathbb{P}(X \in A)$.
4. *La varianza puede simplificar su cálculo como* $var(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$.
5. *El comportamiento de la varianza frente a transformaciones afines es* $var(aX + b) = a^2 \cdot var(X)$.

Además, el siguiente teorema muestra en que sentido la media es la mejor aproximación constante de una variable aleatoria y como la viarainza nos da una idea de su error. Se basa en utilizar como criterio el **error cuadrático medio**.

Propiedades 1.4 $\mathbb{E}[(Y - c)^2] \geq var(Y)$ *con igualdad si y solo si* $c = \mathbb{E}[Y]$.

Demostración 1.1 (Prop. 1.4) *Como demostración, notar que la función a minimizar se puede escribir como una parábola convexa en función de c (usando linealidad de la esperanza):* $\mathbb{E}[Y^2] - 2c\mathbb{E}[Y] + c^2$. *Con lo cuál, su vértice se alcanza en* $c = \mathbb{E}[Y]$, *y para dicho valor el mínimo vale* $\mathbb{E}[(Y - \mathbb{E}[Y])^2] = var(Y)$.

Los vectores aleatorios también tienen sus momentos asociados, definidos a través del operador esperanza. El momento más representativo de un vector (X, Y) es su **covarianza** $\text{cov}(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$. La esperanza aplicadas sobre vectores aleatorios también tiene sus propiedades, algunas de las cuales pueden verse a continuación,

Propiedades 1.5

1. *Al igual que en el caso escalar, sigue valiendo que* $\mathbb{E}[g(X, Y)] = \int_{\mathcal{Y}} \int_{\mathcal{X}} g(x, y)p_{XY}(x, y)dxdy$.
2. *No es necesario conocer la distribución marginal para calcular la media* $\mathbb{E}[X] = \int_{\mathcal{Y}} \int_{\mathcal{X}} x \cdot p_{XY}(x, y)dxdy$.
3. *La linealidad sigue valiendo* $\mathbb{E}[aX + bY + c] = a \cdot \mathbb{E}[X] + b \cdot \mathbb{E}[Y] + c$

4. *La covarianza también puede simplificar su cálculo como* $cov(X,Y) = \mathbb{E}[XY] - \mathbb{E}[X]\cdot\mathbb{E}[Y]$.
5. *La varianza es un caso particular de la covarianza* $cov(X,X) = var(X)$
6. *La covarianza es simétrica* $cov(X,Y) = cov(Y,X)$.
7. *La covarianza es bilineal* $cov(aX+bY+c, \alpha X+\beta Y+\gamma) = a\cdot\alpha\cdot var(X) + b\cdot\beta\cdot var(Y) + (b\cdot\alpha + a\cdot\beta)\cdot cov(X,Y)$.
8. *La varianza de una suma puede calcularse utilizando la covarianza* $var(X+Y) = var(X) + var(Y) + 2\cdot cov(X,Y)$.
9. *El valor esperado de una variable truncada puede calcularse como* $\mathbb{E}[g(X)|X\in A] = \frac{\mathbb{E}[g(X)\cdot\mathbf{1}\{X\in A\}]}{\mathbb{P}(X\in A)}$

Así como la independencia factoriza las distribuciones, también factoriza las esperanzas. Pero la recíproca no es válida. Es decir, si dos variables X e Y son independientes, luego $\mathbb{E}[g(X)h(Y)] = \mathbb{E}[g(X)]\mathbb{E}[h(Y)]$ y por lo tanto $\text{cov}(X,Y) = 0$. Pero descorrelación (covarianza nula) no implica independencia.

Los momentos caracterizan las vectores aleatorios. Así como la Prop. 1.4 demuestra que la constante que mejor aproxima a la variable aleatoria en términos del error cuadrático medio es la media, la recta que mejor aproxima recibe el nombre de **recta de regresión**. La misma es definida a partir de los momentos como se muestra en el siguiente teorema.

Propiedades 1.6 $\mathbb{E}[(Y-(aX+b))^2] \geq var(Y) - \frac{cov(X,Y)^2}{var(X)}$ *con igualdad si y solo si*

$$aX + b = \frac{cov(X,Y)}{var(X)}\,(X - \mathbb{E}[X]) + \mathbb{E}[Y] \tag{1.9}$$

Demostración 1.2 (Prop. 1.6) *Para demostrarlo, notar que la función a minimizar es*

$$\mathbb{E}[Y^2] + a^2\mathbb{E}[X^2] + b^2 - 2a\mathbb{E}[XY] - 2b\mathbb{E}[Y] + 2ab\mathbb{E}[X] \tag{1.10}$$

Para buscar el mínimo, se puede igualar a cero las derivadas respecto de a *y de* b*:*

$$2a\mathbb{E}[X^2] - 2\mathbb{E}[XY] + 2b\mathbb{E}[X] = 2b - 2\mathbb{E}[Y] + 2a\mathbb{E}[X] = 0 \tag{1.11}$$

Luego $b = \mathbb{E}[Y] - a\mathbb{E}[X]$ *y reemplazando*

$$a\mathbb{E}[X^2] - \mathbb{E}[XY] + \mathbb{E}[X]\mathbb{E}[Y] - a\mathbb{E}[X]^2 = 0 \quad \rightarrow \quad a = \frac{cov(X,Y)}{var(X)} \tag{1.12}$$

para esos valores el error cuadrático medio es

$$\mathbb{E}\left[\left(Y - \mathbb{E}[Y] - \frac{cov(X,Y)}{var(X)}(X - \mathbb{E}[X])\right)^2\right]$$

$$= var(Y) + \frac{cov(X,Y)^2}{var(X)} - 2\frac{cov(X,Y)^2}{var(X)} \tag{1.13}$$

$$= var(Y) - \frac{cov(X,Y)^2}{var(X)} \tag{1.14}$$

Los momentos condicionales llevan al máximo el potencial de la esperanza. Sea $\varphi(x) = \mathbb{E}[Y|X = x]$ la media de la distribución de $Y|_{X=x}$ (como función de x) y $\mathbb{E}[Y|X] = \varphi(X)$ a la variable aleatoria construida evaluando a $\varphi(x)$ en la variable aleatoria X. Algunas de las propiedades más importantes pueden verse a continuación.

Propiedades 1.7 *La esperanza y la varianza condicional poseen las siguientes propiedades.*

1. $\mathbb{E}[Y] = \mathbb{E}[\mathbb{E}[Y|X]]$
2. $\mathbb{E}[Yg(X)] = \mathbb{E}[\mathbb{E}[Y|X]g(X)]$
3. $\mathbb{P}(Y \in \mathcal{R}) = \mathbb{E}[\mathbb{P}(Y \in \mathcal{R}|X)]$
4. $var(Y) = \mathbb{E}[var(Y|X)] + var\,(\mathbb{E}[Y|X])$

La característica principal de la esperanza condicional es que es el mejor predictor a la hora de estimar una variable aleatoria como puede verse en el siguiente teorema.

Propiedades 1.8 $\mathbb{E}[(Y - \varphi(X))^2] \geq \mathbb{E}[var(Y|X)]$ *con igualdad si y solo si* $\varphi(x) = \mathbb{E}[Y|X = x]$.

Demostración 1.3 (Prop. 1.8) *La demostración es sencilla: sumando y restando la esperanza condicional en la expresión a minimizar se observa que,*

$$\mathbb{E}[(Y - \varphi(X))^2] = \mathbb{E}[(Y - \mathbb{E}[Y|X] + \mathbb{E}[Y|X] - \varphi(X))^2] \tag{1.15}$$

$$= \mathbb{E}[(Y - \mathbb{E}[Y|X])^2] + \mathbb{E}[(\mathbb{E}[Y|X] - \varphi(X))^2] + 2\mathbb{E}[(Y - \mathbb{E}[Y|X])(\mathbb{E}[Y|X] - \varphi(X))] \tag{1.16}$$

El primer sumando puede simplificando utilizando las propiedades de la esperanza condicional $\mathbb{E}[(Y - \mathbb{E}[Y|X])^2] = \mathbb{E}[\mathbb{E}[(Y - \mathbb{E}[Y|X])^2|X]] = \mathbb{E}[var(Y|X)]$. *El segundo sumando simplemente se acota con* $\mathbb{E}[(\mathbb{E}[Y|X] - \varphi(X))^2] \geq 0$ *y la igualdad se da si y solo si* $\varphi(x) = \mathbb{E}[Y|X = x]$. *El tercer sumando se anula usando las propiedades de la esperanza condicional* $\mathbb{E}[(Y - \mathbb{E}[Y|X])(\mathbb{E}[Y|X] - \varphi(X))] = \mathbb{E}[\mathbb{E}[Y - \mathbb{E}[Y|X]|X](\mathbb{E}[Y|X] - \varphi(X))] = 0$. *Juntando los tres términos el teore-*

ma es probado: $\mathbb{E}[(Y-\varphi(X))^2] \geq \mathbb{E}[var(Y|X)]$ *con igualdad se da si y solo si* $\varphi(x) = \mathbb{E}[Y|X=x]$.

Las propiedades 1.4, 1.6 y 1.8 son esenciales conceptualmente para la inteligencia artificial. Los resultados de estos teoremas implican que:

- En términos del error cuadrático medio, el mejor predictor es la esperanza condicional. Si nos restringimos a las rectas, el predictor óptimo es la recta de regresión. Si en cambio buscamos un predictor constante, la mejor opción es la esperanza.
- Si la esperanza condicional es una recta, necesariamente debe coincidir con la recta de regresión. Si la recta de regresión es una constante debe coincidir con la esperanza $\mathbb{E}[Y]$.
- En ningún caso, el error cuadrático medio puede ser inferior a $\mathbb{E}[\text{var}(Y|X)]$. Esto lo convierte en un límite fundamental.

A continuación se presentará un ejemplo de cómputo de este tipo de magnitudes.

Ejemplo 1.2 *En el mercado de smartphones, los dispositivos con mayor capacidad de almacenamiento suelen tener baterías más duraderas. Modelar estos datos podría ayudar a estimar la duración de la batería en función de su capacidad de almacenamiento, algo útil para los consumidores a la hora de elegir un nuevo dispositivo. Sea* X *la capacidad de almacenamiento de los smartphones (en TB) e* Y *la duración de su batería (en días), con densidad de probabilidad conjunta de la forma:*

$$p_{XY}(x,y) = \frac{3}{4} \cdot \mathbf{1}\left\{0 < y < 1 + x^2, 0 < x < 1\right\} \tag{1.17}$$

Calcular la duración media de las baterías, la recta de regresión y la esperanza condicional. Indicar cuál es el error cuadrático medio asociado a cada aproximación.

Lo primero a determinar es la factorización de la distribución en la condicional $Y|X=x$ (para caracterizar el comportamiento de Y como función de x) y la marginal X (para medir correctamente cuanto se penalizan los errores). Al igual que en el Ej. 1.1, se separará todos los factores donde aparezca y y se completará la densidad para que integre 1 (utilizando la lista de distribuciones conocidas del Cuadro 1.2).

$$p_{XY}(x,y) = \underbrace{\frac{1}{1+x^2}\mathbf{1}\left\{0 < y < 1 + x^2\right\}}_{p_{Y|X=x}(y)} \cdot \underbrace{\frac{3(1+x^2)}{4}\mathbf{1}\left\{0 < x < 1\right\}}_{p_X(x)} \tag{1.18}$$

donde puede verse $Y|_{X=x} \sim \mathcal{U}(0, 1+x^2)$ y X es una variable aleatoria continua con densidad bien determinada, pero sin ser ninguna de las mencionadas en el Cuadro 1.2. La esperanza y varianza condicional pueden observarse en el mencionado cuadro y determinar

que $\mathbb{E}[Y|X=x] = \frac{1+x^2}{2}$ y $\text{var}(Y|X=x) = \frac{(1+x^2)^2}{12}$. El resto del problema es simplemente calcular todos los momentos asociados al problema.

Los momentos de X pueden obtenerse simplemente integrando polinomios

$$\mathbb{E}[X^k] = \int_0^1 x^k \frac{3(1+x^2)}{4} dx = \frac{3}{4}\left(\frac{1}{k+1} + \frac{1}{k+3}\right) \tag{1.19}$$

de esta manera $\mathbb{E}[X] = \frac{9}{16}$, $\mathbb{E}[X^2] = \frac{2}{5}$, $\mathbb{E}[X^3] = \frac{5}{16}$ y $\mathbb{E}[X^4] = \frac{9}{35}$. La varianza será entonces $\text{var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2 = \frac{107}{1280}$. La media de Y se pueden calcular utilizando las propiedades de la esperanza condicional.

$$\mathbb{E}[Y] = \mathbb{E}[\mathbb{E}[Y|X]] = \frac{1+\mathbb{E}[X^2]}{2} = \frac{7}{10} \tag{1.20}$$

Para el caso de la varianza, puede hacerse el mismo tipo de análisis con $\text{var}(Y) = \mathbb{E}[\text{var}(Y|X)] + \text{var}\left(\mathbb{E}[Y|X]\right)$. En este caso

$$\mathbb{E}[\text{var}(Y|X)] = \frac{1+2\mathbb{E}[X^2]+\mathbb{E}[X^4]}{12} = \frac{6}{35} \tag{1.21}$$

$$\text{var}\left(\mathbb{E}[Y|X]\right) = \frac{\text{var}(X^2)}{4} = \frac{\mathbb{E}[X^4] - \mathbb{E}[X^2]^2}{4} = \frac{17}{700} \tag{1.22}$$

y por lo tanto $\text{var}(Y) = \frac{137}{700}$.

Por el lado de la covarianza $\text{cov}(X,Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$, resta calcular la esperanza del producto. La misma puede resolverse utilizando las Prop. 1.7.

$$\mathbb{E}[XY] = \mathbb{E}[X\mathbb{E}[Y|X]] = \frac{\mathbb{E}[X]+\mathbb{E}[X^3]}{2} = \frac{3}{16} \tag{1.23}$$

y por lo tanto $\text{cov}(X,Y) = \frac{7}{160}$. Finalmente utilicemos los resultados 1.4, 1.6 y 1.8, para analizar las diferentes aproximaciones en términos del error cuadrático medio.

- La constante que mejor aproxima a Y es su media $\mathbb{E}[Y] = \frac{7}{10}$, y el error cuadrático medio es $\text{var}(Y) = \frac{137}{700} \approx 0.196$.
- La recta que mejor aproxima a Y es la recta de regresión $\frac{\text{cov}(X,Y)}{\text{var}(X)}(x - \mathbb{E}[X]) + \mathbb{E}[Y] = \frac{56}{107}x + \frac{217}{535}$, y el error cuadrático medio es $\text{var}(Y) - \frac{\text{cov}(X,Y)^2}{\text{var}(X)} = \frac{3236}{18725} \approx 0.173$.
- La función que mejor aproxima a Y es la asociada a la esperanza condicional $\mathbb{E}[Y|X=x] = \frac{1+x^2}{2}$, y el error cuadrático medio es $\mathbb{E}[\text{var}(Y|X)] = \frac{6}{35} \approx 0.171$.

1.2. Estadística

En la mayoría de las aplicaciones, rara vez se conoce la distribución exacta de las variables aleatorias. La estadística soluciona este inconveniente por medio de datos. Si bien el aprendizaje estadístico busca reconocer patrones por medio de ejemplos, su análisis no debe reducirse al conjunto de datos con los que cuenta sino que dichos patrones deben poder generalizarse a nuevas muestras. En ese sentido, la inteligencia artificial es solamente

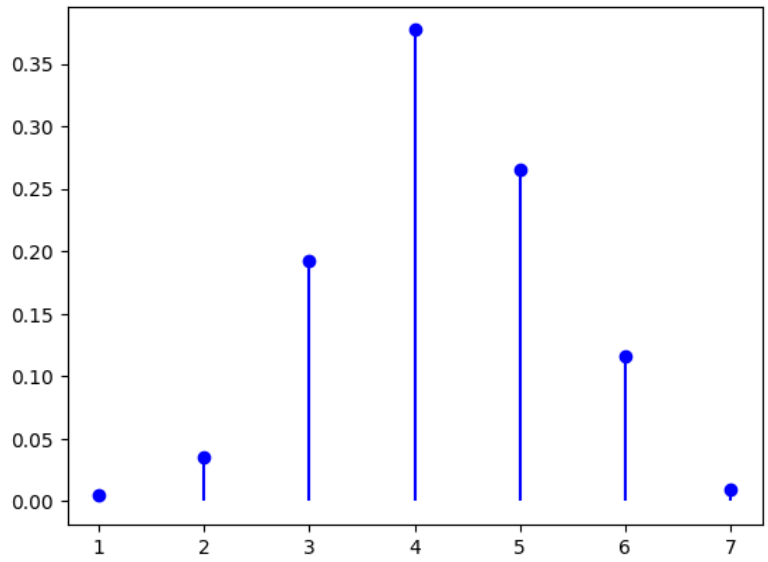


Figura 1.1: Ejemplo de función de masa de probabilidad empírica.

una gran máquina de estadística con suficiente automatización, cálculo computacional y un poco de marketing. Pero no nos pisemos la manguera entre bomberos.

1.2.1. Distribución Empírica e Histograma

La primera pregunta que analizaremos en esta sección es como aproximar una distribución por medio de datos. Uno de los métodos más simples para efectuar dicha aproximación se conoce como **distribución empírica** [2, Capítulo 11]. La distribución empírica asume una distribución discreta, donde la probabilidad de cada átomo corresponde a su frecuencia de aparición $\hat{P}(x) = \frac{K}{n}$ donde n es la cantidad de muestras totales, K la cantidad de muestras con valor x. Un ejemplo de función de masa de probabilidad empírica puede verse en la Fig. 1.1. Su función de distribución asociada será entonces del tipo *escalera*.

Otro enfoque para enfrentar esta problemática es la estimación de densidad por técnicas no paramétricas [3, Capítulo 4], siendo el **histograma** su variante más sencilla. El histograma modela la variable como continua y asume una densidad constante por regiones. En cada región asigna $\hat{p}(x) = \frac{K}{n \cdot V}$ donde n es la cantidad de muestras totales, K la cantidad de muestras en dicha región y V el volumen de la región (en el caso escalar el ancho del intervalo). Este método garantiza que la probabilidad de cada región sea proporcional a la cantidad de muestras que pertenecen a ella. Un ejemplo de función histograma puede verse en la Fig. 1.2.

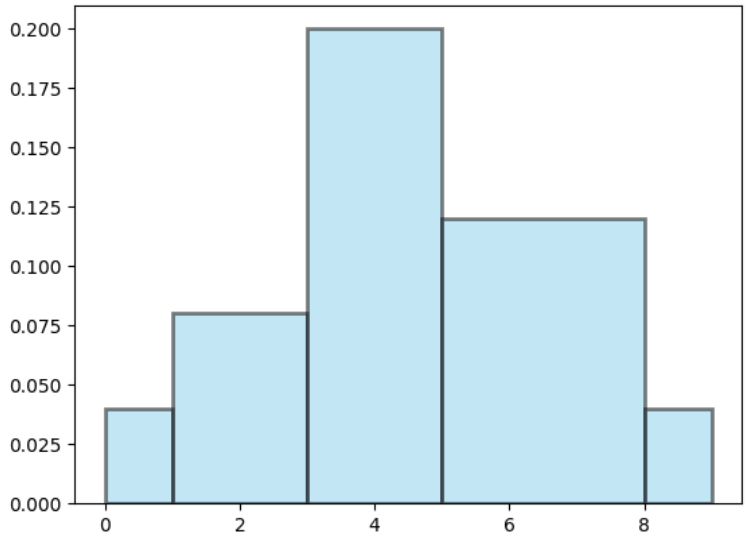


Figura 1.2: Ejemplo de función histograma (densidad de probabilidad).

1.2.2. Simulación

La generación de datos sintéticos es una parte esencial de la estadística. Ya sea para validar los modelos utilizados o para generar nueva información sobre una tarea, la simulación de datos es un área sumamente importante en la temática. En la práctica, cualquier software afín suele tener desarrollados algunos algoritmos generadores de números *pseudo-aleatorios*, pero es razonable que no cuente con todas las distribuciones necesarias que el usuario necesite. El ejemplo más simple es la $\mathcal{U}(0,1)$, que cualquier calculadora científica puede simular. A partir de esos algoritmos, es necesario poder transformar las variables aleatorias generadas para que tengan cualquier distribución deseada. El siguiente teorema muestra algunos de los resultados más importantes en el tema.

Propiedades 1.9 *Para encontrar una transformación que satisfaga una determinada distribución pueden utilizarse los siguientes resultados*

1. *Sea $U \sim \mathcal{U}(0,1)$, luego la variable aleatoria $Y = F_Y^{-1}(U)$ posee función de distribución F_Y, donde F_Y^{-1} es la* inversa generalizada*:*
$$F_Y^{-1}(u) = \text{mín}\{y \in \mathbb{R} : \ u \leq F_Y(y)\} \tag{1.24}$$

2. *Toda variable aleatoria X, con función distribución estrictamente creciente en un intervalo, evaluada en su propia función de distribución $U = F_X(X)$ posee distribución uniforme $U \sim \mathcal{U}(0,1)$.*

3. *Sea X una variable aleatoria con función de distribución estrictamente creciente en un intervalo, luego $Y = F_Y^{-1}(F_X(X))$ posee función de distribución*

F_Y.

La primera propiedad indica como poder transformar variables $\mathcal{U}(0,1)$ en variables con cualquier distribución que se necesite, la segunda explica como poder transformar variables con una determinada distribución en $\mathcal{U}(0,1)$, y la tercera combina los dos resultados anteriores.

En el caso de vectores, si se desea simular un vector aleatorio $(X,Y) \sim p_{XY}$ basta con generar una muestra de $X \sim p_X$, para luego usar dicho valor observado x para generar $Y|X=x \sim p_{Y|X=x}$. En el Ej. 1.1, primero se generarían muestras $X \sim \Gamma(2,1)$ para luego generar muestras $Y|_{X=x} \sim \mathcal{U}(0,x)$ (una y para cada x).

Para simular una variable truncada $X|X \in A$, basta con generar datos de $X \sim p_X$, para luego usar solamente con los que cumplen $x \in A$ (descartando el resto). Para simular un vector truncado $(X,Y)|(X,Y) \in A$, basta con generar datos de $X \sim p_X$, generar sus correspondientes $Y|X=x \sim p_{Y|X=x}$ y luego quedarme con los pares $(x,y) \in A$.

1.2.3. Estadística Frecuentista

El objetivo de reconocer y estimar toda la distribución de una variable aleatoria por medio de datos es sumamente ambicioso. En la mayoría de los casos no se cuenta con una cantidad de datos suficiente para tal objetivo. Es entonces cuando surge la necesidad de incorporar supuestos previos sobre las variables involucradas, **modelando** el problema estadístico. No es necesario que dichos supuestos sean totalmente ciertos, sino que basta con que balanceen el desempeño esperado, la complejidad del modelo y la cantidad de datos con las que se cuentan.

El modelado consiste en asumir información parcial en el conocimiento de la distribución de la variable $p(x|\theta)$ [4]. Es decir que se conoce dicha distribución exceptuando un conjunto de parámetros $\theta \in \Theta$. La estadística frecuentista asume que las muestras son independientes e idénticamente distribuidas para cada posible parámetro, y a la distribución conjunta del set de datos observados $\mathbf{x} = (x_1, \cdots, x_n)$ se la conoce como **verosimilitud**:

$$\mathcal{L}(\theta) = p(\mathbf{x}|\theta) = \prod_{i=1}^{n} p(x_i|\theta) \tag{1.25}$$

La **bondad de un estimador** está dada por la relación de compromiso sesgo/varianza, como se puede ver en el siguiente teorema.

Propiedades 1.10 $\mathbb{E}[(\hat{\theta}-\theta)^2|\theta] = var(\hat{\theta}|\theta) + \mathcal{B}^2(\theta)$, *donde* $\mathcal{B}(\theta) = \theta - \mathbb{E}[\hat{\theta}|\theta]$ *se denomina sesgo.*

La demostración puede verse al final de la sección. La relación de compromiso sesgo/varianza explica que un buen estimador necesita tener simultáneamente bajo sesgo y

baja varianza. Se habla de relación de compromiso porque muchas de las soluciones más utilizadas para mejorar uno de los términos termina perjudicando al otro.

Uno de los estimadores puntuales más utilizados en la estadística frecuentista, debido a su consistencia, es el **estimador de máxima verosimilitud** $\hat{\theta}_{\text{MV}} = \arg\max_{\theta \in \Theta} \mathcal{L}(\theta)$, es decir elegir los parámetros que maximicen la verosimilitud (los estimadores son funciones de la muestra observada). Bajo ciertas condiciones de regularidad dicha estimación puede efectuarse igualando a cero la derivada del logaritmo[5] de la verosimiltud:

$$\sum_{i=1}^{n} \frac{\partial}{\partial \theta} \log p(x_i|\hat{\theta}_{\text{MV}}) = 0 \tag{1.26}$$

donde $\log(\cdot)$ hace referencia al *logaritmo natural*. Una vez caracterizados los parámetros, el modelo es capaz de predecir el comportamiento de nuevas muestras no observadas. Sea x_{test} una muestra no observada en $\mathbf{x}$, las predicciones se efectuan a través del **principio de invarianza**: la estimación por máxima verosimilitud de cualquier función de θ puede calcularse evaluando dicha función en $\hat{\theta}_{\text{MV}}$ [5].

$$\widehat{p}_{\text{MV}}(x_{\text{test}}) = p(x_{\text{test}}|\hat{\theta}_{\text{MV}}) \tag{1.27}$$

Demostración 1.4 (Prop. 1.10) *El error cuadrático medio puede descomponerse como:*

$$\mathbb{E}[(\hat{\theta} - \theta)^2|\theta] = \mathbb{E}[(\hat{\theta} - \mathbb{E}[\hat{\theta}|\theta] + \mathbb{E}[\hat{\theta}|\theta] - \theta)^2|\theta] \tag{1.28}$$

$$= \mathbb{E}[(\hat{\theta} - \mathbb{E}[\hat{\theta}|\theta])^2|\theta] + (\mathbb{E}[\hat{\theta}|\theta] - \theta)^2 + 2\mathbb{E}[(\hat{\theta} - \mathbb{E}[\hat{\theta}|\theta])|\theta](\mathbb{E}[\hat{\theta}|\theta] - \theta) \tag{1.29}$$

donde cada uno de los sumandos se puede simplificar como:

- $\mathbb{E}[(\hat{\theta} - \mathbb{E}[\hat{\theta}|\theta])^2|\theta] = var(\hat{\theta}|\theta)$
- $(\mathbb{E}[\hat{\theta}|\theta] - \theta)^2 = \mathcal{B}^2(\theta)$
- $\mathbb{E}[(\hat{\theta} - \mathbb{E}[\hat{\theta}|\theta])|\theta](\mathbb{E}[\hat{\theta}|\theta] - \theta) = 0$

Reemplazando en la expresión correspondiente el teorema fue demostrado.

1.2.4. Estadística Bayesiana

La estadística bayesiana busca verdades en contexto de incertidumbre, interpretando la probabilidad como una medida de credibilidad en un evento [6, Capítulo 1]. El modelo no solo representa el fenómeno a predecir, sino también nuestra propia ignorancia sobre el mismo. Esto no quiere decir que las ciencias empíricas están condenadas a decir "no se" a todas las hipótesis que uno desea contrastar, sino que busca evitar mentir maximizando incertidumbre (no afirmar más de lo que se sabe) dada la información disponible (sin

[5]Por ser una función monótona, no modifica la ubicación de los máximos.

ocultar lo que efectivamente se sabe).

A nivel técnico, la estadística bayesiana representa los parámetros del modelo por medio de una variable aleatoria T con distribución *a priori* $p_T(\theta)$ [7]. En este tipo de modelos, la hipótesis de independencia es válida *cuando se conoce el parámetro*. Es decir que la verosimilitud de una muestra puede escribirse como $p_{\mathbf{X}|T=\theta}(\mathbf{x}) = \prod_{i=1}^{n} p_{X|T=\theta}(x_i)$. No se pierde generalidad en asumir que las variables son idénticamente distribuidas[6].

El corazón de la estadística bayesiana es la *distribución a posteriori*, la cuál se deduce por medio de la **regla de bayes** combinando la distribución *a priori* con la verosimilitud.

$$p_{T|\mathbf{X}=\mathbf{x}}(\theta) \propto p_T(\theta) \cdot \prod_{i=1}^{n} p_{X|T=\theta}(x_i) \tag{1.30}$$

La distribución *a posteriori* nos permite definir estimadores puntuales a partir de ella. En el caso de buscar parámetros dentro de un conjunto Θ discreto, se suele elegir como estimador el **máximo a posteriori** $\hat{\theta}_{\text{MAP}} = \arg\max_{\theta \in \Theta} p_{T|\mathbf{X}=\mathbf{x}}(\theta)$. En el caso de Θ continuo, la elección habitual suele ser la **media a posteriori** $\hat{\theta}_{\text{BAY}} = \mathbb{E}[T|\mathbf{X} = \mathbf{x}]$.

Sin embargo, el verdadero potencial de la estadística bayesiana radica en hacer predicciones sin necesidad de estimadores puntuales. En este sentido, este tipo de estadística no solo puede resolver los mismos problemas que la frecuentista, sino que también pueden intentar resolver problemas donde la estadística clásica es insuficiente o iluminar el sistema subyacente con un modelado más flexible. Es entonces que se define la **distribución predictiva**.

$$p_{X_{\text{test}}|\mathbf{X}=\mathbf{x}}(x_{\text{test}}) = \int_{\Theta} p_{X|T=\theta}(x_{\text{test}}) p_{T|\mathbf{X}=\mathbf{x}}(\theta) d\theta \tag{1.31}$$

donde X_{test} es una variable aleatoria no vista en el conjunto de entrenamiento $\mathbf{X}$. A continuación se analizará un ejemplo mostrando como trabajar con este tipo de estadística analíticamente.

Ejemplo 1.3 *El tiempo de vida (en años) de un transistor es una variable aleatoria con distribución exponencial de parámetro θ. A priori se modela θ como una variable aleatoria con distribución $\Gamma(2, 3)$. Si en 20 transistores se observó una duración total $\sum_{i=1}^{20} x_i = 7$.*

1. *Hallar la distribución a posteriori del parámetro θ.*
2. *Hallar la distribución predictiva del tiempo de vida de un transistor.*

Como primer paso en un problema bayesiano, hay que comenzar planteando la distribución

[6] Se podría haber escrito $p_{X_i|T=\theta}(x_i)$ en su lugar, pero no es necesario

a posteriori. En este caso evitaremos las constantes de proporcionalidad:

$$p_{T|\mathbf{X}=\mathbf{x}}(\theta) \propto p_T(\theta) \cdot \prod_{i=1}^{n} p_{X|T=\theta}(x_i) \propto \theta e^{-3\theta} \mathbf{1}\{\theta > 0\} \cdot \prod_{i=1}^{20} \theta e^{-\theta x_i} = \theta^{21} e^{-10\theta} \mathbf{1}\{\theta > 0\} \quad (1.32)$$

Es decir, la variable se distribuye *a posteriori* como $T|_{\mathbf{X}=\mathbf{x}} \sim \Gamma(22, 10)$. La distribución predictiva es de la forma

$$p_{X_{\text{test}}|\mathbf{X}=\mathbf{x}}(x_{\text{test}}) = \int_{\Theta} p_{X|T=\theta}(x_{\text{test}}) p_{T|\mathbf{X}=\mathbf{x}}(\theta) d\theta \propto \int_0^{\infty} \theta e^{-\theta x_{\text{test}}} \mathbf{1}\{x_{\text{test}} > 0\} \cdot \theta^{21} e^{-10\theta} d\theta \quad (1.33)$$

Reconociendo el núcleo de la integral, se puede observar que el mismo es proporcional a la densidad de una $\Gamma(\nu, \lambda)$, es decir $p(x) = \frac{\lambda^\nu}{\Gamma(\nu)} x^{\nu-1} e^{-\lambda x} \mathbf{1}\{x > 0\}$. Sabiendo que por ser densidad debe integrar 1:

$$p_{X_{\text{test}}|\mathbf{X}=\mathbf{x}}(x_{\text{test}}) \propto \int_0^{\infty} \theta^{22} e^{-\theta(10+x_{\text{test}})} d\theta \cdot \mathbf{1}\{x_{\text{test}} > 0\} \propto \frac{1}{(10 + x_{\text{test}})^{23}} \mathbf{1}\{x_{\text{test}} > 0\} \quad (1.34)$$

donde se utilizó $\nu = 23$ y $\lambda = 10 + x_{\text{test}}$. Esta distribución se la conoce como Lomax (véase Cuadro 1.2) $X_{\text{test}}|_{\mathbf{X}=\mathbf{x}} \sim \text{Lomax}(22, 10)$.

Tanto *a priori* como *a posteriori*, la variable T es una Gamma. Este fenómeno de mantenerse dentro de una familia ocurre por cierta compatibilidad entre la distribución *a priori* y la verosimilitud (en este caso una exponencial, caso particular de la Gamma). Cuando se da este fenómeno se dice que la distribución *a priori* es una **conjugada a priori**. Las soluciones analíticas suelen proponer conjugadas, como distribución *a priori*, ya que así garantizan que la distribución *a posteriori* pertenezca a una familia conocida (la misma que la distribución *a priori*). Es simplemente una recomendación para hacer sencillos (o al menos factibles) los cálculos.

1.2.4.1. Estadísticos Suficientes

Un concepto muy útil a la hora de efectuar inferencia es el de **estadístico suficiente**. Un estadístico $S(\mathbf{X})$ se denomina suficiente para θ si la distribución de $\mathbf{X}|_{S(\mathbf{X})=s}$ no depende de θ. Es decir que toda la información que posee la muestra sobre θ se encuentra en el estadístico. Además, el teorema de Neyman-Fisher nos permite encontrar estadísticos suficientes de forma muy sencilla [4, Capítulo 6].

Propiedades 1.11 (Teorema de Neyman-Fisher) *El estadístico $S(\mathbf{X})$ es suficiente, si y solo si su verosimilitud se puede descomponer como:*

$$p_{\mathbf{X}|T=\theta}(\mathbf{x}) = g(\theta, S(\mathbf{x})) \cdot h(\mathbf{x}) \quad (1.35)$$

En términos bayesianos un estadístico suficiente se interpreta como una independencia condicional $\mathbf{X} \perp \theta|_{S(\mathbf{X})=s}$ (es decir que la muestra y los parámetros son independientes cuando se conoce el estadístico suficiente). Este resultado implica que la distribución *a posteriori* debe cumplir $p_{T|\mathbf{X}=\mathbf{x}}(\theta) = p_{T|S(\mathbf{X})=s(\mathbf{x})}(\theta)$, y por lo tanto nos permite intercambiar

el conocimiento de toda la muestra por el del estadístico suficiente. En el ejemplo anterior la distribución solo dependía de la muestra a través de la suma, estadístico suficiente para θ en una distribución exponencial.

Esto nos permite hacer equivalencias sobre los datos de las variables observadas. En el Ej. 1.3, es equivalente pensar que se cuenta con 20 muestras $\exp(\theta)$ que con una sola muestra $\Gamma(20, \theta)$[7]. Otro ejemplo clásico donde se da este fenómeno es en las variables Bernoulli, donde también la suma es estadístico suficiente: es equivalente tener n muestras $\text{Ber}(p)$ que una muestra $\text{Bin}(n, p)$.

1.2.4.2. Test de hipótesis

TODO

[7]La suma de 20 variables $\exp(\theta)$ independientes e idénticamente distribuidas se distribuye como una $\Gamma(20, \theta)$

2

Regresión en Inteligencia Artificial

Tal vez el mayor desafío no sea aprender a usar la inteligencia artificial, sino redefinir nuestro aporte humano en un mundo que automatiza incluso la inteligencia.

El objetivo de la inteligencia artificial es resolver tareas de forma automatizada y con la mejor calidad posible. Ésta tecnología está modificando de forma acelerada la matriz laboral tal como la conocíamos. Tareas que antes requerían horas de redacción, análisis o asistencia técnica ahora pueden ser resueltas en minutos por una inteligencia artificial entrenada para comprender y generar material con una sorprendente fluidez. Esta transformación no se limita a sectores creativos o administrativos; también está empezando a influir en el desarrollo de software, la ingeniería de datos y la automatización de procesos, áreas donde muchos ingenieros se formaron creyendo que la demanda sería estable o creciente.

Esto plantea una pregunta incómoda pero necesaria: ¿qué lugar ocuparemos los profesionales en un entorno donde las máquinas no solo ejecutan, sino también piensan -al menos en términos funcionales-? El rol del ingeniero ya no se limita a diseñar sistemas eficientes, sino que debe integrar consideraciones éticas, adaptarse a herramientas inteligentes y desarrollar una visión crítica sobre la tecnología que crea. Tal vez el mayor desafío no sea aprender a usar estos modelos, sino redefinir nuestro aporte humano en un mundo que automatiza incluso la inteligencia.

En cualquier problema de **aprendizaje supervisado**, es decir inferir la **etiqueta** Y a partir del **predictor** X[1], siempre hay algunas magnitudes que se pueden destacar. El objetivo siempre será minimizar el valor esperado de la llamada **función costo** $\ell(x, y)$, el cuál recibe el nombre de **riesgo esperado** $\mathbb{E}[\ell(X, Y)]$. La estimación que minimice dicho riesgo se conocerá como **solución óptima** y llamaremos **error bayesiano** al mínimo error posible capaz de ser alcanzado (asociado a la solución óptima). Estamos hablando de un límite fundamental para el error que nunca podrá ser mejorado independientemente de la tecnología utilizada.

En particular, en este capítulo estudiaremos el **problema de regresión**, es decir esti-

[1] Véase el Capítulo 4 para detalles precisos sobre el término.

mar $Y = \varphi(X)$ utilizando el error cuadrático como función costo $\ell(x, y) = (y - \varphi(x))^2$ (y por lo tanto el error cuadrático medio como riesgo esperado). En el capítulo anterior se demostró, Prop. 1.8, que la regresión óptima es $\mathbb{E}[Y|X = x]$ y con ésta el error alcanza el error bayesiano $\mathbb{E}[\text{var}(Y|X)]$. Todo este resultado es la base del aprendizaje estadístico: lo mejor que puedo hacer es utilizar la esperanza condicional como regresor y el menor error al que puedo aspirar es el bayesiano.

La inteligencia artificial lejos está de terminarse con este resultado. En la práctica, el problema radica en no conocer la distribución de los datos, y por lo tanto no poder calcular fidedignamente la esperanza condicional. El aprendizaje estadístico propone entonces **aprender** la esperanza condicional por medio de datos.

2.1. Relación de Compromiso Sesgo/Varianza

La solución inmediata que uno puede proponer al problema de regresión es la **minimización del riesgo empírico**. Es decir, encontrar la función $\varphi(x)$ que minimice $\frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} \ell(x_i, y_i)$ para un conjunto de datos observado $\{(x_i, y_i)\}_{i=1}^{n_{\text{tr}}}$. El problema está en que el verdadero objetivo es minimizar el riesgo esperado $\mathbb{E}[\ell(X, Y)]$ que no necesariamente va a coincidir con el empírico. En este sentido surge el **gap de generalización**: la capacidad de generalizar el comportamiento de los datos observados a datos desconocidos (representados por los valores esperados).

$$\underbrace{\mathbb{E}[\ell(X,Y)]}_{\text{Riesgo esperado}} = \underbrace{\frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} \ell(x_i, y_i)}_{\text{Riesgo empírico}} + \left(\underbrace{\mathbb{E}[\ell(X,Y)] - \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} \ell(x_i, y_i)}_{\text{Gap de generalización}} \right) \tag{2.1}$$

En este sentido, para disminuir el riesgo esperado se necesita simultáneamente tratar de minimizar el riesgo empírico y el gap de generalización; dos magnitudes de características muy distinta [8, Sección 2.9]. Este problema es análogo al estimador paramétrico puntual estudiado en la Prop. 1.10: es una relación de compromiso entre el sesgo (representado por el riesgo empírico) y la varianza (representado por el gap).

Cuando hablamos de errores cuadráticos, hay que tener en consideración que este tipo de error es una magnitud difícil de interpretar por no tener valor máximo. Si bien tener un error nulo implica desempeño perfecto, otro valor de error requiere contextualizarlo para catalogarlo como insuficiente o satisfactorio. En este sentido, el gap de generalización propone un marco interpretativo al ser una diferencia: ¿Que tanto más grande es el riesgo esperado con respecto al empírico? En cambio el riesgo empírico por si solo no es interpretable. Para darle un sentido se lo compara con el error bayesiano, ya que este es el error al que todo algoritmo desea alcanzar. En la práctica el error bayesiano suele ser

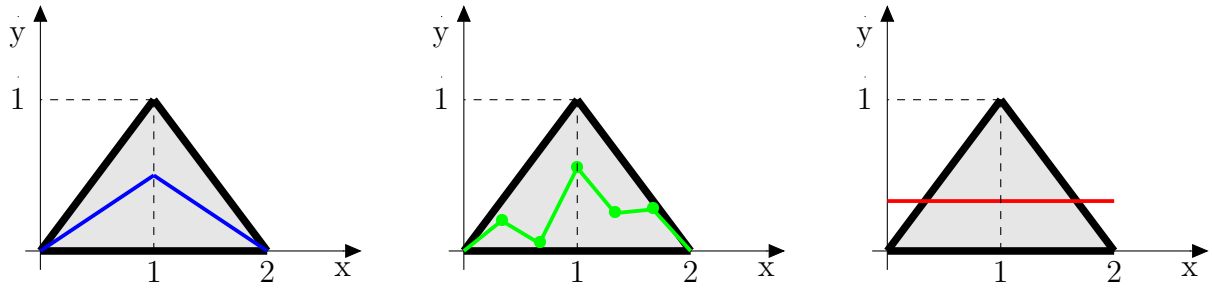


Figura 2.1: Regresores para el Ej. 2.1. (a) Solución $\mathbb{E}[Y|X=x]$ asociada a la esperanza condicional, (b) solución empírica y (c) recta de regresión.

livianamente estimado con imaginación: ¿Que error creo que se puede llegar a alcanzar en esta tarea? En muchos casos, el *error humano* es buen candidato.

A continuación se analizará un ejemplo para entender las diferencias entre los riesgos empírico y esperado.

Ejemplo 2.1 *Sea (X,Y) un vector uniforme en el triángulo de vértices $(0,0)$, $(1,1)$ y $(2,0)$. Analizar posibles regresores para este problema.*

Comencemos analizando la solución óptima. Al factorizar la distribución conjunta uniforme, se observa una condicional uniforme (véase Ej. 1.2). En este caso,

$$Y|X=x \sim \begin{cases} \mathcal{U}(0,x) & \text{Si } 0<x<1 \\ \mathcal{U}(0,2-x) & \text{Si } 1<x<2 \end{cases} \tag{2.2}$$

y por lo tanto, solución $\mathbb{E}[Y|X=x]$ asociada a la esperanza condicional puede verse gráficamente en la Fig. 2.1a. Analíticamente, tanto la esperanza como la varianza condicional se definen a partir del Cuadro 1.2:

$$\mathbb{E}[Y|X=x] = \begin{cases} \frac{x}{2} & \text{Si } 0<x<1 \\ \frac{2-x}{2} & \text{Si } 1<x<2 \end{cases} \qquad \text{var}(Y|X=x) = \begin{cases} \frac{x^2}{12} & \text{Si } 0<x<1 \\ \frac{(2-x)^2}{12} & \text{Si } 1<x<2 \end{cases} \tag{2.3}$$

En este caso el error bayesiano se puede calcular utilizando que el triángulo tiene área unitaria:

$$\mathbb{E}[\text{var}(Y|X)] = \int_0^1 \int_0^x \frac{x^2}{12} dydx + \int_1^2 \int_0^{2-x} \frac{(2-x)^2}{12} dydx = \frac{1}{24} \tag{2.4}$$

Esas son la solución ideal y el mínimo error esperado al que se puede aspirar. Sin embargo, dado un conjunto de datos $\{(x_i,y_i)\}_{i=1}^{n_{\text{tr}}}$, cuando se elige una solución minimizando el riesgo empírico se encuentran regresores como el visto en la Fig. 2.1b. Esta solución no solo minimiza $\frac{1}{n_{\text{tr}}}\sum_{i=1}^{n_{\text{tr}}} \ell(x_i,y_i)$ sino que directamente alcanza error nulo. Sin embargo,

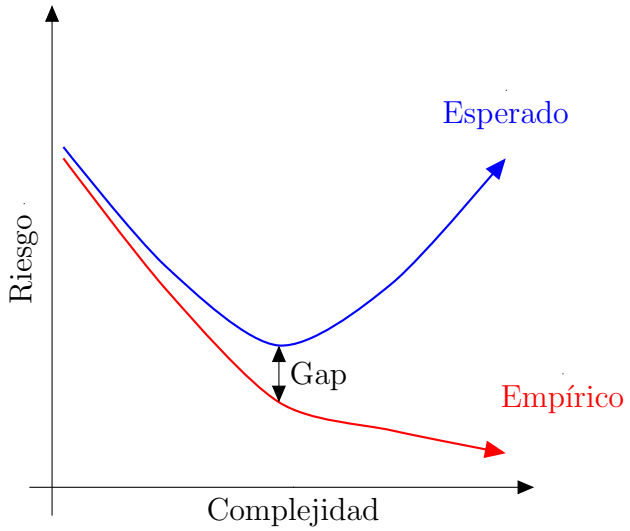


Figura 2.2: Relación de compromiso sesgo/varianza típica según la teoría clásica de generalización.

soluciones de este tipo suelen poseer un algo gap de generalización debido a un exceso de complejidad en el modelado, es decir un **problema de varianza**. En este caso, al darle libertad total a la elección del regresor, se eligió un regresor mucho más complejo que el óptimo $\mathbb{E}[Y|X = x]$. Las soluciones que poseen un excesivo problema de varianza se dice que tienen **overfitting**, ya que éstas se sobreajustan a los datos. Se suele interpretar este tipo de soluciones como algoritmos que **memorizan** en lugar de aprender.

En cambio, si uno asigna una complejidad excesivamente baja al regresor se encuentra con un **problema de sesgo**. En la Fig. 2.1c se postula la recta de regresión como posible regresor con este tipo de problema. Dentro de los regresores de complejidad lineal, la recta de regresión es el óptimo (véase Prop 1.6). Posiblemente esta solución tenga un muy bajo gap de generalización (ni siquiera se calculó utilizando los datos), sin embargo al tener una complejidad mucho más baja que la solución óptima, el riesgo empírico será importante. Las soluciones que poseen un excesivo problema de sesgo se dice que tienen **underfitting**, ya que éstas se subajustan a los datos.

La Fig. 2.2 muestra como la teoría clásica de generalización caracteriza esta relación de compromiso. Los modelos de muy baja complejidad ven imposibilitado cualquier tipo de sobreajuste y tratarán de forma similar los datos conocidos a los desconocidos (alcanzando un bajo gap de generalización). En contraposición, los modelos muy complejos pueden reducir el riesgo empírico tanto como quieren pero corren el riesgo de sobreajustar. Encontrar el balance óptimo, lejos de ser trivial, es problema principal de la inteligencia artificial.

La cantidad de datos con la que se cuenta juega un rol vital en este análisis. Si se

cuenta con la posibilidad de obtener más y más datos llegará un momento en que un modelo, de una complejidad determinada, no podrá sobreajustarlos. Bajo ciertas hipótesis de consistencia, el gap de generalización deberá disminuir con el número de muestras (en la medida que los promedios tiendan a los valores esperados en (2.1))[2]. En este sentido, aumentar la cantidad de datos soluciona problemas de varianza, pero no así de sesgo (al tener más muestras, es más difícil representarlas a todas con una complejidad fija). En cualquier caso, se deberá adaptar la complejidad del modelo a la cantidad de datos con las que se cuenta, pudiendo permitirse modelos más complejos cuando se cuenta con grandes cantidades de datos y limitando la misma cuando la cantidad es escueta.

2.2. Regresión Lineal

El objetivo del aprendizaje estadístico es intentar disminuir simultáneamente el riesgo empírico y el gap de generalización para así reducir el riesgo esperado (2.1). Pero mientras que el gap de generalización no se conoce, el riesgo empírico es totalmente computable y por lo tanto podemos detectar fácilmente cuando estamos en presencia de un problema de sesgo. La idea de la regresión lineal es muy sencilla: limitar al máximo la complejidad del modelo (soluciones lineales) para tener un gap de generalización moderado (evitar problemas de varianza) y posteriormente verificar si hay problema de sesgo. En caso de que no los haya podemos concluir que tenemos un aceptable riesgo esperado. En este sentido, la regresión lineal busca aprender la recta de regresión[3] (véase Prop. 1.6) en lugar de la esperanza condicional, básicamente porque es una tarea mucho más sencilla[4].

Sea $\{(x_i, y_i)\}_{i=1}^{n_{\text{tr}}}$ un conjunto de datos con $x_i \in \mathbb{R}^{d_x}$ e $y_i \in \mathbb{R}$. Se propone buscar soluciones de la forma $\hat{y}(x) = w^T x + b$ con $w \in \mathbb{R}^{d_x}$ y $b \in \mathbb{R}$ minimizando el riesgo empírico. Es decir, un algoritmo de inteligencia artificial tiene dos etapas bien diferenciadas: una primera llamada **entrenamiento** donde se calcularán los parámetros (w y b en este caso) y una segunda etapa llamada **predicción** donde a cada x se le asignará un $\hat{y}(x)$ (en este caso $\hat{y}(x) = w^T x + b$). En el caso de la regresión lineal, el entrenamiento entonces buscará calcular

$$(w, b) = \underset{w \in \mathbb{R}^{d_x}, b \in \mathbb{R}}{\arg\min} \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} (w^T \cdot x_i + b - y_i)^2 \tag{2.5}$$

Esta ecuación define un problema de proyección ortogonal de álgebra lineal. Esto se puede analizar **vectorizando** la ecuación. Definiendo $\mathbf{X} \in \mathbb{R}^{n_{\text{tr}} \times (d_x+1)}$, $\mathbf{y} \in \mathbb{R}^{n_{\text{tr}}}$ y $\mathbf{w} \in$

[2]Dado que el regresor elegido depende del mismo conjunto de datos con el que se mide el riesgo empírico, el análisis es sofisticado (la *ley de los grandes números* no es suficiente para explicar el fenómeno). Para más información ver [7, Sección 6.5]

[3]Para predictores unidimensionales es una recta, en general son hiperplanos.

[4]Será una recta si X es escalar, un plano si tiene dos dimensiones, etc. En general será un hiperplano.

$\mathbb{R}^{d_x+1}$ como

$$\mathbf{X} = \begin{pmatrix} 1 & x_1^T \\ 1 & x_2^T \\ \vdots & \vdots \\ 1 & x_{n_{\text{tr}}}^T \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n_{\text{tr}}} \end{pmatrix}, \qquad \mathbf{w} = \begin{pmatrix} b \\ w \end{pmatrix} \tag{2.6}$$

el problema (2.5) puede reducirse a minimizar $J(\mathbf{w}) = \frac{1}{n_{\text{tr}}}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$. Para ello basta con analizar la primera derivada (gradiente) y la segunda derivada (matriz Hessiana) respecto a $\mathbf{w}$ (para más información sobre derivadas respecto vectores/matrices ver [9]).

$$\nabla J(\mathbf{w}) = \frac{2}{n_{\text{tr}}}\mathbf{X}^T(\mathbf{X}\mathbf{w} - \mathbf{y}), \qquad \mathcal{H}_J(\mathbf{w}) = \frac{2}{n_{\text{tr}}}\mathbf{X}^T\mathbf{X} \tag{2.7}$$

Es habitual en los problemas de regresión que $n_{\text{tr}} \gg d_x$, lo cuál se suele traducir en una matriz Hessiana inversible (teniendo en cuenta que los x fueron elegidos de forma aleatoria). En ese caso, será también una matriz definida positiva y por lo tanto el problema será convexo. Es decir que el resultado obtenido de igualar a cero el gradiente de $J(\mathbf{w})$ será efectivamente un mínimo. Se puede despejar entonces para encontrar el procedimiento del entrenamiento:

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{2.8}$$

La solución es la **pseudoinversa** de la matriz $\mathbf{X}$ multiplicada por $\mathbf{y}$. A continuación se presenterá un ejemplo de como funciona la regresión lineal.

Ejemplo 2.2 *Se desea hacer una regresión lineal (sin normalizar) sobre el siguiente conjunto de datos:*

x	*0.2*	*1.4*	*-1.4*	*-0.2*
y	*20.0*	*10.0*	*10.0*	*0.0*

- *Hallar los parámetros del modelo.*
- *Predecir y para $x = 0.1$.*

Basta con definir las matrices de (2.8):

$$\mathbf{X}^T\mathbf{X} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0.2 & 1.4 & -1.4 & -0.2 \end{pmatrix} \begin{pmatrix} 1 & 0.2 \\ 1 & 1.4 \\ 1 & -1.4 \\ 1 & -0.2 \end{pmatrix} = \begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix} \tag{2.9}$$

$$\mathbf{w} = \begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix}^{-1} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0.2 & 1.4 & -1.4 & -0.2 \end{pmatrix} \begin{pmatrix} 20 \\ 10 \\ 10 \\ 0 \end{pmatrix} = \begin{pmatrix} 10 \\ 1 \end{pmatrix} \tag{2.10}$$

con lo cual $b = 10$ y $w = 1$; y finalmente $\hat{y}(0.1) = 10.1$.

2.2.1. Codificación de variables categóricas

La regresión lineal, así como la mayor parte de los algoritmo de inteligencia artificial, requieren que los valores de sus entradas tengan valores numéricos. Pero hay determinados tipos de variables, donde no es posible encontrar una **relación de orden**, donde las mismas simplemente representa categorías (con una cantidad finita de opciones posibles)[5].

Por ejemplo, podemos contar con una base de datos donde una de sus columnas represente el color de un objeto. Supongamos que los resultados posibles son *rojo, verde, azul* y *negro*. Si asignamos respectivamente los valores 0, 1, 2 y 3 estaríamos diciendo que el rojo está mas cerca del verde que del negro, lo cuál sesgaría nuestro análisis por ser falso.

Para evitar este tipo de decisiones arbitrarias, se suele codificar a las variables categóricas sin relación de orden específica con representaciones **One-Hot**. Cada columna categórica, de K clases posibles, se convierte en K variables binarias donde siempre una y solamente una de ellas toma el valor 1. En nuestro ejemplo, codificaríamos el *rojo* con $(1, 0, 0, 0)$, el *verde* con $(0, 1, 0, 0)$, el *azul* con $(0, 0, 1, 0)$ y el *negro* con $(0, 0, 0, 1)$ (es decir 4 variables en total). Notar que dos colores cualesquiera distintos siempre están a una distancia geométrica de $\sqrt{2}$.

Es importante destacar que el proceso de codificación se define *durante el entrenamiento*. Es decir, que si al momento de efectuar una predicción se le solicita al algoritmo una categoría no vista anteriormente, se le suele asignar a todas las variables codificadas el valor 1. De esta manera simultáneamente tendrá una distancia constante al resto de las categorías ($\sqrt{K-1}$), y una categoría válida tendrá más cerca al resto de categorías válidas que a esta codificación particular. *One-Hot* permite procesar muy fácilmente las variables categóricas, pero puede aumentar considerablemente la cantidad de variables de entrada o **predictores**.

2.3. Gradiente Descendente

La regresión lineal tiene la ventaja de contar con solución analítica (2.8), pero la mayoría de las funciones a minimizar no poseen esa ventaja. Además si tenemos en cuenta que para un d_x muy grande la inversa de la matriz presente en (2.8) no es computable, nos damos cuenta de la necesidad de contar con un método numérico para minimizar funciones.

El método del **gradiente descendente** es un algoritmo numérico de optimización presentado por Cauchy muchos años atrás [10] y, sin embargo, es la esencia de la mayoría de los algoritmos modernos de inteligencia artificial. La idea es sencilla: igualar a cero la

[5]En las variables que solo pueden tomar dos valores, la relación de orden es intrascendente. En esta sección nos referimos a variables con mayor cantidad de valores posibles.

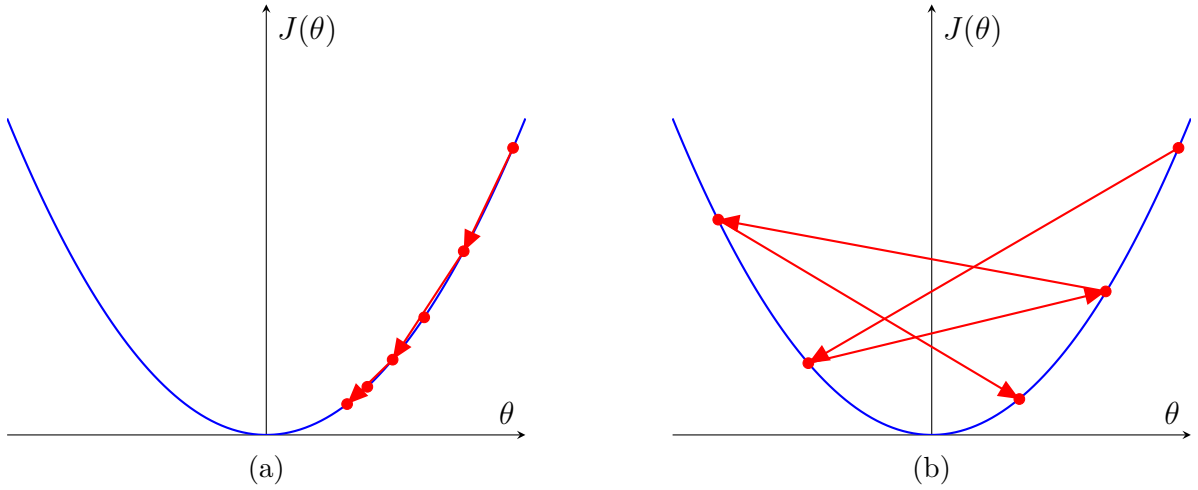


Figura 2.3: Comparación del comportamiento del gradiente descendente con distinto *learning rate*. (a) caso convergente y (b) caso divergente.

derivada de una función a minimizar $J(\theta)$ *numéricamente*. Es decir, avanzar *poco a poco* (de forma iterativa) en la dirección del máximo decrecimiento de la función.

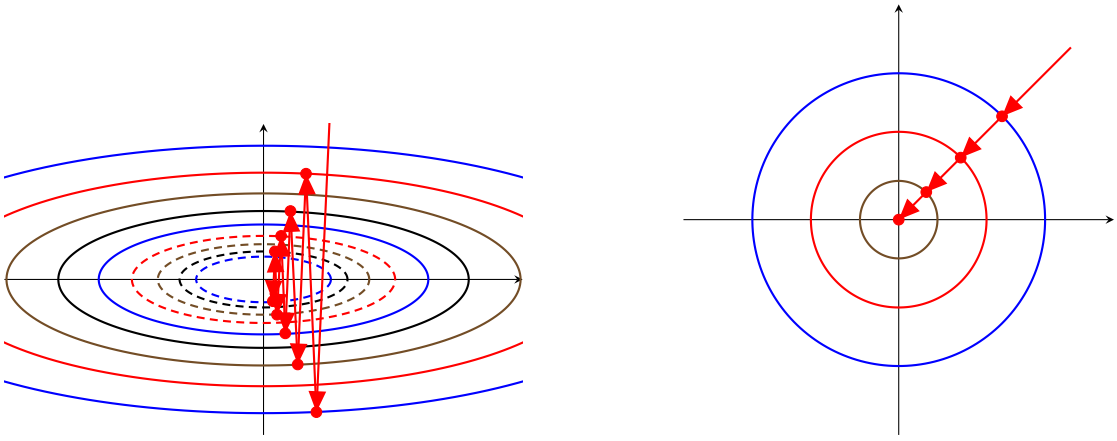
$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla J(\theta_{t+1}) \tag{2.11}$$

donde $\alpha > 0$ recibe el nombre de **learning rate** o tasa de aprendizaje. Este tipo de parámteros que no se deciden durante el entrenamiento reciben el nombre de **hiper-parámetro**, para diferenciarlos de los parámetros entrenables. Según el valor de α, el comportamiento del algoritmo puede ser bien distinto. En la Fig. 2.3a se puede observar un ejemplo convergente del algoritmo. Paso a paso el algoritmo se va acercando al mínimo, aunque corre el riesgo de necesitar muchas iteraciones para alcanzar la convergencia. Sin embargo un *learning rate* muy grande, lejos de acelerar, puede generar comportamientos divergentes en el algoritmo como puede verse en la Fig. 2.3b.

Por desgracia no existe un optimizador universal que funcione para cualquier tarea y conjunto de datos [11], dependerá de en cada problema el encontrar un valor de α adecuado. En la práctica se suele apuntar al valor más grande que genere un comportamiento convergente, eligiéndolo por *prueba y error*. A continuación analizaremos algunos detalles a tener en cuenta a la hora de utilizar este algoritmo.

2.3.1. Normalización como pre-procesamiento

El gradiente descendente, descripto en (2.11), tiene la característica en tener un mismo valor de α para todas las direcciones. El motivo de esto, tiene que ver con solamente tener que seleccionar un hiper-parámetro en lugar de muchos. Sin embargo, puede que no exista un valor que satisfaga a todas las direcciones. Es por esto que surge la necesidad



(a) (b)

Figura 2.4: Comparación del gradiente descendente (a) sin y (b) con normalización de las variables de entrada a partir de las curvas de nivel.

de **normalizar**.

La normalización de cada componente permite poner a todas las variables en la misma unidad. Esta fuerza a todas las variables de entrada o *predcitores* a tener valor medio nulo y varianza unitaria (empíricamente hablando). Formalmente asigna los valores

$$(\mathbf{x})_k \leftarrow \frac{(\mathbf{x})_k - \mu_k}{\sigma_k} \tag{2.12}$$

donde las μ_k y σ_k son calculadas previo al entrenamiento como:

$$\mu_k = \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} (\mathbf{x}_i)_k, \qquad \sigma_k = \sqrt{\frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} \left[(\mathbf{x}_i)_k - \mu_k\right]^2} \tag{2.13}$$

con n_{tr} la cantidad de muestras de entrenamiento. La Fig. 2.4a muestra el comportamiento de la minimización por gradiente descendente sobre una superficie representada por las curvas de nivel, denotando un comportamiento errático (recordar que el gradiente es ortogonal a las curvas de nivel [12, Capítulo 13]). En contraste, la Fig. 2.4b muestra su contraparte normalizada, donde con menos iteraciones se logró optimizar los parámetros. Cabe destacar que la normalización se define durante la *etapa de entrenamiento*, fijando los valores de μ_k y σ_k. A la hora de realizar una predicción, se utilizará la normalización ya calculada previamente.

El uso de la normalización no se restringe solamente a algoritmos optimizados por gradiente descendente. Además de solucionar posibles problemas de convergencia, permite forzar media nula en los predictores hipótesis de muchos métodos basados en álgebra lineal del aprendizaje automático. También permite que sea pertinente la relación entre los predictores. Supongamos por ejemplo que contamos con un conjunto de datos que posee como variables la superficie de una vivienda a cotizar y como otra variable la cantidad de habitaciones. No debería ser distinto decir que una vivienda posee $36m^2$ y 3 habitaciones

(12 veces más grande un número que el otro) que $600cm^2$ y 3 habitaciones (200 veces más grande). En ese sentido es muy útil cuando las magnitudes involucradas tienen diferentes unidades.

Sin embargo, vale resaltar que hay que tener muy claro el motivo de la normalización. Ya sea para ayudar la convergencia, para forzar media nula o para volver las magnitudes comparables, es necesario entender por que se utiliza. Normalizar preventivamente cualquier conjunto de datos suele traer muchos problemas.

2.3.2. Learning Rate óptimo

En esta sección se buscarán garantías teóricas de optimalidad para α. Esto es solamente un análisis teórico, en la práctica el *learning rate* se suele elegir intentando tomar el máximo valor posible convergente. Esto está relacionado con asociar el α con la velocidad de convergencia, lo cuál en rigor de verdad no es totalmente cierto. El presente análisis teórico nos permitirá entender las limitaciones de este tipo de asociaciones. Se estudiarán garantías sobre la velocidad de convergencia para un algoritmo de gradiente descendente sobre una función $J(\theta)$ genérica (no necesariamente regresión lineal, pero sin perderla de vista), aunque con un mínimo de hipótesis razonables de convexidad:

- Existe un único θ^* tal que $\nabla J(\theta^*) = 0$.
- La matriz Hessiana $\mathcal{H}_J(\theta)$ existe y es definida positiva para todo θ.

A continuación se utlizarán algunos resultados matemáticos conocidos. En primer lugar, se reescribirá $\nabla J(\theta_t)$ utilizando el **teorema de Taylor** de primer orden, alrededor del mencionado θ^* [13, Apéndice A.6]:

$$\nabla J(\theta_t) = \nabla J(\theta^*) + \mathcal{H}_J(\tilde{\theta}) \cdot (\theta_t - \theta^*) \tag{2.14}$$

para algún $\tilde{\theta}$ en el segmento que une θ_t y θ^*. Notar que $\mathcal{H}_J(\tilde{\theta})$ es una matriz real, cuadrada y simétrica. Por lo tanto, por el **teorema espectral** [14, Sección 4.1], puede escribirse como $\mathcal{H}_J(\tilde{\theta}) = Q^T \Lambda Q$ con una matriz de autovalores Λ diagonal y una de autovectores Q ortogonal $Q^T Q = Q Q^T = I$. Utilizando este resultado y que $\nabla J(\theta^*) = 0$ se obtiene $\nabla J(\theta_t) = Q^T \Lambda Q(\theta_t - \theta^*)$. En general, Λ y Q podrán ser desconocidas (aunque en el caso de regresión lineal pueden calcularse ya que $\mathcal{H}_J(\tilde{\mathbf{w}}) = \frac{2}{n_{\text{tr}}} \mathbf{X}^T \mathbf{X}$ no depende de $\tilde{\mathbf{w}}$). Con el fin de estudiar la recurrencia, se reescribirá la diferencia $(\theta_{t+1} - \theta^*)$ utilizando la definición del gradiente descendente (2.11):

$$\theta_{t+1} - \theta^* = \theta_t - \theta^* - \alpha \nabla J(\theta_t) \tag{2.15}$$

$$= \left(I - \alpha Q^T \Lambda Q\right)(\theta_t - \theta^*) \tag{2.16}$$

$$= Q^T (I - \alpha \Lambda) Q (\theta_t - \theta^*) \tag{2.17}$$

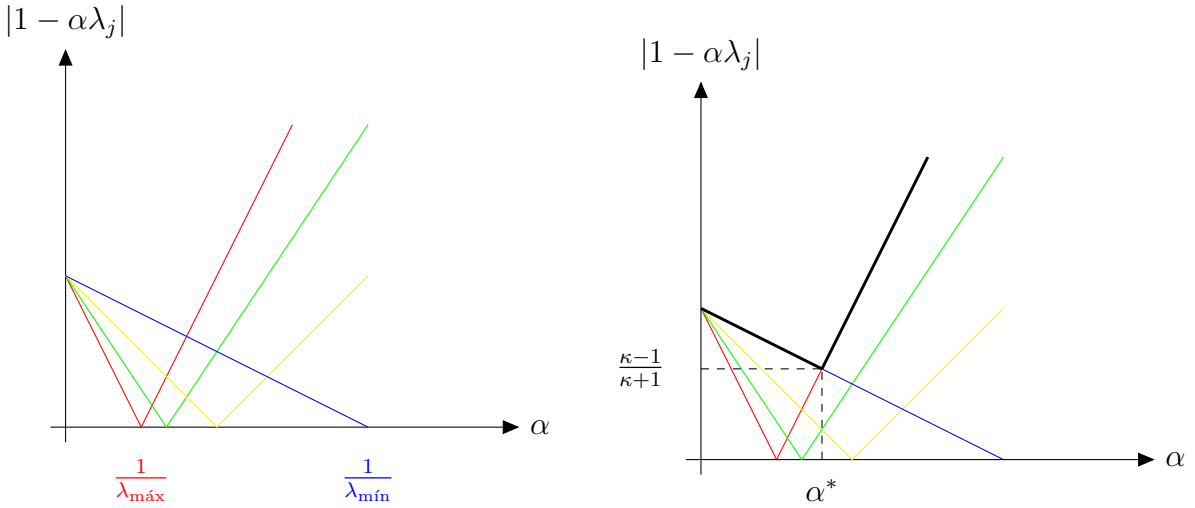


Figura 2.5: Resolución gráfica del problema (2.18). En (a) se resaltan los valores $|1-\alpha\lambda_j|$ para los diferentes autovalores λ_j; y en (b) se resalta la resolución del problema minmax.

Sea $v_t = Q\left(\theta_t - \theta^*\right)$, la relación de recurrencia (2.17) puede escribirse como $v_{t+1} = \left(I - \alpha\Lambda\right) v_t$ y por lo tanto $v_t = \left(I - \alpha\Lambda\right)^t v_0$. Hay una relación directa entre la convergencia en θ_t y la convergencia en v_t, por lo que para estudiar garantías, ésta última es suficiente. Como criterio de garantía sobre la velocidad de convergencia se elegirá un criterio de peor caso:

$$\min_{\alpha} \max_{j} |1-\alpha\lambda_j| \quad \text{s.t.} \quad |1-\alpha\lambda_j| < 1 \quad \forall\, j \tag{2.18}$$

donde λ_j son los elementos de la diagonal de Λ. Es un problema **minmax** con restricciones. Por el lado de las restricciones, la matriz $(I - \alpha\Lambda)$ es naturalmente diagonal y la convergencia estará dada cuando cada coeficiente sea menor que uno en valor absoluto (por estar elevada a la cantidad de iteraciones). Como criterio de garantía de velocidad se elige minimizar el peor caso de éstos (su máximo). Es importante notar que el α obtenido de esta manera no será el mejor posible en cada caso, sino el que me brinda garantías.

Este problema minmax se puede resolver gráficamente [15, Capítulo 9]. En la Fig. 2.5a se muestran los los diferentes valores $|1-\alpha\lambda_j|$ como función de α. Cada curva en V obtiene su mínimo cuando $\alpha = \frac{1}{\lambda_j}$, y habrá un valor mínimo y un valor máximo (serán positivos porque son autovalores de una matriz definida positiva). El máximo valor de estas curvas se puede ver en la Fig. 2.5b: la máxima curva comienza siendo la de menor autovalor ($\lambda_{\text{mín}}$), hasta que se cruza con la curva de mayor autovalor ($\lambda_{\text{máx}}$). En el vértice, o mínimo valor de esta nueva curva, se encuentra justamente el *learning rate* óptimo α^*. Para encontrar el valor hace falta intersectar las dos curvas: la semirrecta decreciente de la curva con mínimo autovalor con la semirrecta creciente de la curva con máximo autovalor.

$$1 - \alpha^*\lambda_{\text{mín}} = \alpha^*\lambda_{\text{máx}} - 1 \quad \rightarrow \quad \alpha^* = \frac{2}{\lambda_{\text{mín}} + \lambda_{\text{máx}}} \tag{2.19}$$

La velocidad de convergencia estará asociada entonces al valor correspondiente a este *learning rate* en el eje de las ordenadas:

$$1 - \frac{2}{\lambda_{\text{mín}} + \lambda_{\text{máx}}}\lambda_{\text{mín}} = \frac{\lambda_{\text{máx}} - \lambda_{\text{mín}}}{\lambda_{\text{máx}} + \lambda_{\text{mín}}} = \frac{\kappa - 1}{\kappa + 1} \tag{2.20}$$

donde $\kappa = \frac{\lambda_{\text{máx}}}{\lambda_{\text{mín}}}$ se denomina **número de condición** de la matriz asociada $\mathcal{H}_J(\tilde{\theta})$. El *learning rate* óptimo no coincide con el máximo valor convergente; la condición $|1 - \alpha\lambda_j| < 1$ para todo j se puede reescribir como $0 < \alpha < \frac{2}{\lambda_j}$ y por lo tanto la condición de convergencia es $\alpha < \frac{2}{\lambda_{\text{máx}}}$. Un α muy grande convergente, puede implicar rebotes a la hora de converger como es el caso del ejemplo de la Fig. 2.4a.

Vale la pena volver a mencionar que este análsis es teórico. En regresión lineal, donde se conoce la matriz $\mathcal{H}_J(\tilde{\mathbf{w}}) = \frac{2}{n_{\text{tr}}}\mathbf{X}^T\mathbf{X}$, no se suele utilizar el valor α^* principalmente porque calcular los autovalores es tan costoso como invertir la matriz para utilizar (2.8).

2.4. Regresión Polinómica

En la Sección 2.2 se presentó la *regresión lineal* como una solución que garantiza baja complejidad; en caso de alcanzar un riesgo empírico bajo hay ciertas garantías de buen desempeño. El problema surge cuando el riesgo empírico no alcanza un valor suficientemente satisfactorio, denotando que la complejidad del modelo es insuficiente.

El objetivo más ambicioso, en un problema de regresión, es estimar la regresión asociada a la esperanza condicional $\mathbb{E}[Y|X = x]$, ya que esta minimiza el *error cuadrático medio* (Prop. 1.8). Independientemente de la complejidad de la esperanza condicional, el **teorema de Taylor** indica que cualquier función se puede aproximar como una combinación lineal de coeficientes polinómicos. Es entonces que surge la **regresión polinómica**; utilizar una aproximación lineal sobre un **mapa polinómico** de los *predictores* vectorizándolos de la siguiente manera:

$$\mathbf{X} = \begin{pmatrix} 1 & x_{1,1} & x_{1,2} & x_{1,1}^2 & x_{1,2}^2 & x_{1,1}x_{1,2} \\ 1 & x_{2,1} & x_{2,2} & x_{2,1}^2 & x_{2,2}^2 & x_{2,1}x_{2,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n_{\text{tr}},1} & x_{n_{\text{tr}},2} & x_{n_{\text{tr}},1}^2 & x_{n_{\text{tr}},2}^2 & x_{n_{\text{tr}},1}x_{n_{\text{tr}},2} \end{pmatrix} \tag{2.21}$$

donde $x_{i,k}$ es la muestra i-ésima de la variable k-ésima. El ejemplo presentado en (2.21) corresponde a un *mapa polinómico* de orden 2. En general, el mapa polinómico de orden ν contendrá todos los productos cruzados de las variables hasta orden ν inclusive.

Propiedades 2.1 *La vectorización correspondiente a un mapa polinómico de orden ν sobre d predictores posee una cantidad de columnas $\binom{d+\nu}{\nu}$.*

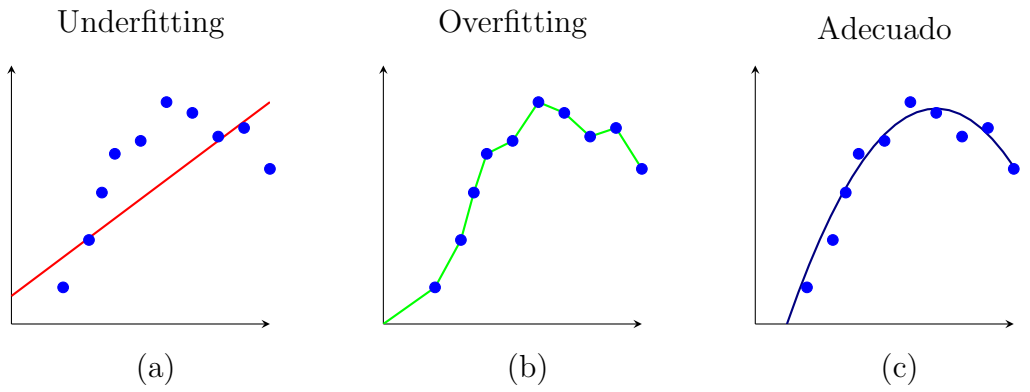


Figura 2.6: Comparación entre regresores denotando (a) un ejemplo de subajuste, (b) uno de sobreajuste, y (c) uno con un ajuste razonable.

Demostración 2.1 (Prop. 2.1) *Los coeficientes de la vectorización de un mapa polinómico pueden escribirse como*

$$1^{a_0} \cdot \prod_{k=1}^{d} x_{i,k}^{a_k} \quad \text{para algunos} \quad a_k \in \mathbb{N}_0, \quad \sum_{k=0}^{d} a_k = \nu \tag{2.22}$$

donde los a_k efectúan una **codificación**. *Para determinar la cantidad de columnas del mapa, basta con resolver el problema combinatorio de cuantas maneras posibles se pueden elegir los a_k. En (2.21), las codificaciones son de la forma $(2,0,0)$, $(1,1,0)$, $(1,0,1)$, $(0,2,0)$, $(0,0,2)$ y $(0,1,1)$. Para resolver el problema en general, es conveniente estudiarlo como un problema de conteo de objetos indistinguibles. Se cuenta con ν puntos para repartir en $k+1$ recipientes (por lo tanto es necesario k separadores). Por ejemplo, el $(1,1,0)$ de (2.21) será representado como "$\times|\times|$" y el $(0,2,0)$ como "$|\times\times|$". Es decir que de las $\nu+d$ posiciones se eligen ν para colocar la $\times$ obteniendo así $\binom{d+\nu}{\nu}$ columnas.*

Es importante destacar que el mapa polinómico entrega predictores en magnitudes no comparables. Por ejemplo, si una variable estaba medida en m, se incorporarán nuevas variables medidas en m^2, m^3, etc. Por ese motivo es indispensable luego de utilizar un mapa polinómico normalizar[6]. El siguiente ejemplo ilustra esta idea.

Ejemplo 2.3 *Preprocesar el siguiente conjunto de datos, para iniciar el entrenamiento de una regresión polinómica de orden 2. ¿Que cantidad de parámetros tendrá el modelo?*

[6]La columna de unos no se normaliza Esta solamente se incorpora como requisito para efectuar posteriormente la regresión lineal (para que exista el término constante).

x_1	*1.2*	*0.8*	*1.0*
x_2	*2.3*	*1.3*	*1.6*

Un mapa polinómico de orden 2 con 2 variables generará un mapa de $\binom{4}{2} = 6$ columnas (y por lo tanto tendrá 6 parámetros la regresión lineal posterior). Las columnas serán: la comunas de unos, las columnas x_1, x_2, x_1^2, x_2^2 y x_1x_2. Calculando las columnas restantes se obtiene

x_1	x_2	x_1^2	x_2^2	x_1x_2
1.2	2.3	1.44	5.29	2.76
0.8	1.3	0.64	1.69	1.04
1.0	1.6	1.0	2.56	1.6

El siguiente paso es normalizar, ya que el mapa polinómico genera variables incomparables en términos de unidad. Calculando la media y el desvío de cada columna, utilizando (2.13), se obtiene:

	x_1	x_2	x_1^2	x_2^2	x_1x_2
μ	1.0	1.73	1.03	3.18	1.8
σ	0.16	0.42	0.33	1.53	0.72

Aplicando la normalización $\frac{x-\mu}{\sigma}$ e incorporando la columna de unos se obtiene el mapa pedido.

bias	x_1	x_2	x_1^2	x_2^2	x_1x_2
1	1.22	1.35	1.26	1.38	1.34
1	-1.22	-1.03	-1.18	-0.97	-1.06
1	0.0	-0.32	-0.08	-0.4	-0.28

El problema de cambiar la regresión lineal por la polinómica es que se pierde la característica de baja complejidad, pudiendo así tener problemas de *overfitting* como se mencionó en la Sección 2.1. En la Fig. 2.6 se muestran ejemplos de regresores, mostrando posibles problemas tanto de subajuste (complejidad insuficiente) como sobreajuste (complejidad excesiva). Recordando (2.1), mientras que el *underfitting* lo podemos detectar por

un alto riesgo empírico el *overfitting* requiere conocer el *gap de generalización*, magnitud que no puede conocerse de forma exacta por depender de valores esperados. Es entonces que surge la necesidad de tener diferentes conjuntos de datos.

2.4.1. Conjuntos de datos

En general se necesitan datos para efectuar distintas funciones: para entrenar, para ajustar la relación de compromiso sesgo/varianza y simplemente para estimar el riesgo esperado.

- **Conjunto de entrenamiento** (*train set*): Datos utilizados para minimizar el riesgo empírico. Sobre estos se produce el "aprendizaje". Las variables definidas a partir de este conjunto se llaman parámetros.
- **Conjunto de validación** (*validation or development set*): Datos utilizados para comparar modelos. Las variables definidas a partir de este conjunto (o definidas previas al entrenamiento) se llaman hiper-parámetros.
- **Conjunto de testeo** (*test set*): Datos utilizados para evaluar el desempeño final del algoritmo. Su única función es presentar estimadores insesgados de las métricas de error y no es imprescindible.

En el caso de una regresión polinómica básica, se pueden entrenar varios modelos para diferentes valores de ν (cada uno de ellos entrenado con el conjunto de entrenamiento) y se elegirá el valor de ν que minimice el error medido con el conjunto de validación. Una vez elegido el ν y teniendo el modelo entrenado para ese valor, se procede a medir el error con el conjunto de testeo (estimando así el riesgo esperado) para evaluar si la decisión fue o no satisfactoria. El mismo procedimiento se puede efectuar para elegir el valor del *learning rate* α o cualquier otro hiper-parámetro involucrado.

En la bibliografía y en la documentación muchas veces se habla de conjunto de testeo en general para referirse a todos los datos no usados durante el entrenamiento. Pero es importante tener en consideración las diferencias entre validación y testeo: la validación se utiliza para tomar decisiones sobre el diseño del modelo, eligiendo hiper-parámetros. Una vez que se utilizó dicho conjunto de datos para tomar una decisión, las estimaciones de error dejan de ser insesgadas: como se utilizaron los datos de validación para elegir el modelo, analizar el desempeño del algoritmo con ellos mismo puede dejar de ser representativo. Sin embargo, si la validación no es minuciosa, puede considerarse aceptable utilizarla como métrica de error (aunque sesgada, la estimación puede ser suficiente). Contar con el conjunto de testeo puede ser prescindible, su única función es medir la eficacia

final del sistema. Es recomendable usarlo principalmente luego de validaciones exhaustivas o cuando no se efectuaron validaciones en absoluto (y es razonable en desconfiar en el desempeño del sistema). Incluso en el caso de contar con un solo predictor, se puede analizar el *overfitting/underfitting* de un algoritmo gráficamente sin necesidad de contar con el conjunto de testeo, como es el caso de la Fig. 2.6.

El problema con generar los diferentes conjuntos de datos es que disminuye la cantidad de datos usado durante el entrenamiento. No hay una regla para asignar proporciones a estos conjuntos, dependerá de cuantos datos se tenga, de la complejidad del modelo y de la dificultad de la tarea. Por ejemplo en el caso de regresión lineal, basta con tener solamente un conjunto de entrenamiento, ya que se asume que el gap de generalización es pequeño por la complejidad del modelo. En lugar de validar el resultado, simplemente analizo si se alcanzó o no un razonable riesgo empírico.

2.4.2. Regularización

El *problema de sesgo* se detecta cuando el riesgo empírico de entrenamiento es grande comparado con el supuesto error bayesiano. El mismo se soluciona aumentando la complejidad del modelo. En cambio, *problema de varianza* se detecta cuando el riesgo empírico de validación es grande comparado con el de entrenamiento, y la mejor manera de combatirlo es aumentando la cantidad de muestras. Por desgracia esto no siempre es posible, ya sea por la dificultad de obtener los datos o de procesarlos. Las técnicas destinadas a combatir el *overfitting* sin incorporar nuevos datos se denominan **regularización**.

Existen diferentes técnicas de regularización: generar datos sintéticos para incorporarlos, incorporar ruido a las muestras para dificultar el sobreajuste, limitar la complejidad del modelo, entre otras. Pero quizás la más importante es el agregado de un término de penalización al riesgo a minimizar durante el entrenamiento. Básicamente se busca perturbar la optimización del modelo, minimizando en lugar del riesgo empírico, el **riesgo regularizado**:

$$J(\theta) = \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} \ell(x_i, y_i) + \lambda \cdot R(\theta) \tag{2.23}$$

donde $\lambda \geq 0$ es un hiper-parámetro que controlará el *overfitting*, y $R(\theta)$ recibe el nombre de **regularizador**. El resigo regularizado define una función a optimizar durante el entrenamiento, rara vez es utilizado en la etapa de testeo. La incorporación del regularizador tiene por objeto acercar la función a optimizar $J(\theta)$ al riesgo esperado (2.1). En ese sentido, un buen regularizador será aquel donde $\lambda R(\theta)$ sea representativo del *gap de generalización*. Cuando $\lambda = 0$ la regularización es ignorada, mientras que para λ muy grandes lo que es ignorado son los datos.

El regularizador más utilizado en regresión polinómica es el denominado **L2**, *weight*

decay o *regularización de Tikhonov*: $R(w,b) = \frac{1}{n_{\text{tr}}}\|w\|^2$. Una primera interpretación de esta selección es que el incorporar la norma cuadrática de los pesos w en la función a minimizar tenderá a "apagar" coeficientes $w_j \approx 0$, y por lo tanto la complejidad efectiva del modelo bajará, tendiendo a disminuir el *overfitting*. Otra posible interpretación es que limitando el valor de w lo que estamos haciendo es tendiendo a limitar el máximo valor posible de la función costo $\ell(x,y) \leq L$, y por lo tanto limitando el *gap de generalización* y por lo tanto el *overfitting*.

$$\mathbb{E}[\ell(X,Y)] - \frac{1}{n_{\text{tr}}}\sum_{i=1}^{n_{\text{tr}}} \ell(x_i,y_i) \leq L \tag{2.24}$$

En la práctica, en regresión polinómica se suele dejar un ν fijo dándole un margen suficiente de complejidad al modelo y controlar el overfitting con el hiper-parámetro de regularización. Es decir que la etapa de validación se efectuará sobre λ en lugar de ν. Esto es beneficioso porque permite un incremento controlado de la regularización ya que mientras $\lambda \in \mathbb{R}$, $\nu \in \mathbb{N}$ necesita dar saltos discretos.

Ejemplo 2.4 *Hallar una solución matricial al problema de regresión lineal sin sesgo y con regularización L2. Analizar el comportamiento para algoritmos no regularizados y muy regularizados.*

El riesgo regularizado para este problema es de la forma

$$J(w) = \frac{1}{n_{\text{tr}}}\|\mathbf{X}w - \mathbf{y}\|^2 + \frac{\lambda}{n_{\text{tr}}}\|w\|^2 \tag{2.25}$$

Basta con analizar la primera derivada (gradiente) y la segunda derivada (matriz Hessiana) respecto a w (para más información sobre derivadas respecto vectores/matrices ver [9]).

$$\nabla J(w) = \frac{2}{n_{\text{tr}}}\mathbf{X}^T(\mathbf{X}w - \mathbf{y}) + \frac{2\lambda}{n_{\text{tr}}}w, \qquad \mathcal{H}_J(w) = \frac{2}{n_{\text{tr}}}(\mathbf{X}^T\mathbf{X} + \lambda \cdot \mathbf{I}) \tag{2.26}$$

donde la matriz Hessiana es claramente definida positiva (y por lo tanto inversible). Esto implica que el problema es convexo y por lo tanto igualando a cero el gradiente equivale a minimizar el riesgo regularizado. Se puede despejar entonces:

$$w = (\mathbf{X}^T\mathbf{X} + \lambda \cdot \mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{2.27}$$

Por un lado, si $\lambda = 0$ se obtiene como solución $w = \mathbf{X}^\dagger\mathbf{y}$ donde $\mathbf{X}^\dagger = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ es la *pseudoinversa*. Por el otro si λ es muy grande (lo suficiente para que $\lambda \cdot \mathbf{I}$ enmascare a $\mathbf{X}^T\mathbf{X}$ pero no tanto para que $w \approx 0$) se obtiene $w \approx \frac{1}{\lambda}\mathbf{X}^T\mathbf{y}$. Esto quiere decir que un algoritmo regularizado interpreta la transpuesta como la operación inversa. Esta operación tiene un muy bajo costo computacional por lo que es utilizada para inversión de problemas físicos bajo el nombre de *Linear Back Projection* [16].

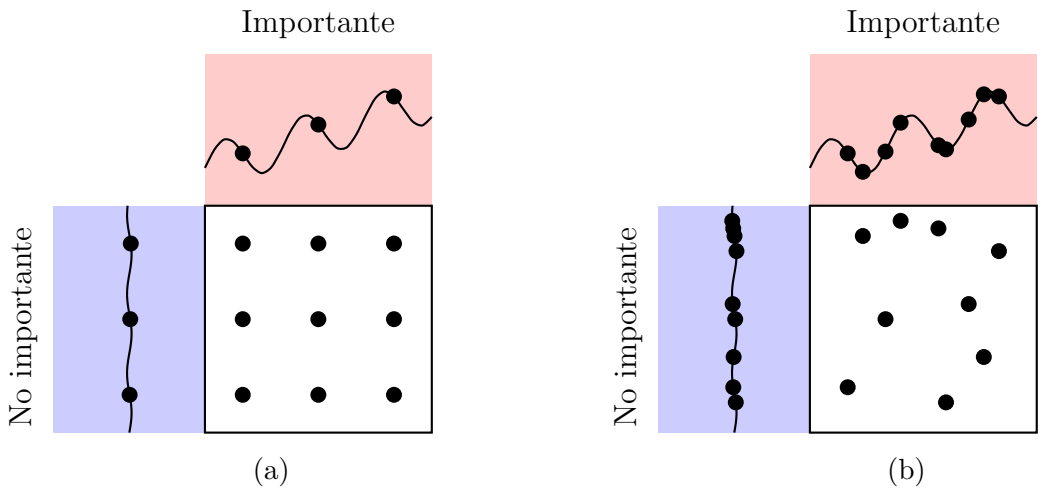


Figura 2.7: Ejemplos de grilla para validar dos hiper-parámetros, de los cuales solo uno importa fuertemente sobre el error. (a) Una grilla regular, y (b) una grilla aleatoria.

2.4.3. Etapa de validación

La etapa de validación está basada en el enfoque de *prueba error*. Por ejemplo, si se desea validar el hiperparámetro λ se deberán realizar V entrenamientos diferentes, donde V es la cantidad de puntos de λ que deseo probar. Finalmente se termina decidiendo por el valor de λ que menor error de validación genere. El problema principal de este enfoque aparese cuando se desea validar varios hiper-parámetros. Por ejemplo, si queremos probar V valores de λ y V valores de α, necesitaríamos realizar V^2 simulaciones. La etapa de validación es computacionalmente costosa, y es necesario tener en cuenta algunas consideraciones para no sufrir tanto esta particularidad. La cantidad de entrenamientos a efectuar crece exponencialmente con la cantidad de hiper-parámetros a validar, fenómeno que recibe el nombre de **maldición de la dimensionalidad**. Este fenómeno está presente también con las muestras de entrenamiento: La necesidad de muestras crece exponencialmente con la cantidad de predictores.

Supongamos que se desean validar dos hiper-parámetros, probando V valores para cada uno y efectuando V^2 entrenamientos en total. En el caso de que solamente uno de los hiper-parámetros sea influyente sobre el error (a priori esta información era desconocida), se terminan realizando V^2 entrenamientos para probar solo V valores del hiper-parámetro relevante. Este fenómeno puede verse en la Fig. 2.7a, donde en los márgenes se muestra una curva de como varía el error de validación con los respectivos hiper-parámetros. En este ejemplo, vemos que se debieron realizar 9 entrenamientos para probar efectivamente solo 3 valores del hiper-parámetro relevante.

Es por este motivo que al validar dos o más hiper-parámetros se opta por efectuar una grilla aleatoria, como se muestra en la Fig. 2.7b. Al elegir valores al azar, es muy

probable que no haya valores repetidos y por lo tanto se terminen probando V^2 valores del hiper-parámetro relevante. En este ejemplo, observando el margen superior, se puede ver que se probaron hiper-parámetros con errores más bajos.

Otra recomendación a la hora de validar parámetros por grilla aleatoria es efectuarla por etapas. Por ejemplo, si cuento con la posibilidad de realizar 300 entrenamientos, quizás sea conveniente primero validar con 100 de ellos. Con esos resultados uno puede limitar la zona donde el error de validación deba ser más chico, y hacer una segunda búsqueda (con otros 100 entrenamientos) dentro de ese espacio limitado. Repitiendo por tercera vez con los 100 entrenamientos restantes logré concentrar muchos valores al rededor de la zona de mayor interés.

Por último, hay muchos casos donde la cantidad de datos observados es limitada y no es posible reservarse un conjunto de validación. Es entonces cuando surgen técnicas un poco más sofisticadas.

2.4.3.1. Validación Cruzada

La etapa de validación es crítica en modelos de alta complejidad, ya que los hiper-parámetros asociados con la regularización suelen ser muy sensibles. Este es el caso de λ en regresión polinómica, el cuál suele necesitar ser validado. El problema es que definir un conjunto de datos de validación, como se mencionó en la Sección 2.4.1, muchas veces es prohibitivo debido a que no se cuenta con suficientes datos. Sin embargo, existen algunas técnicas para validar hiper-parámetros sin necesidad de definir un conjunto de validación, repitiendo el entrenamiento en múltiples ocasiones (y por lo tanto pagando un costo computaional). Estas técnicas se conocen como **validación cruzada**.

Leave-one-out cross-validation (LOOCV) es el método básico de validación cruzada. Propone reservar una sola muestra para validación y entrenar con el resto. Este proceso se repite para cada muestra, realizando n_{tr} entrenamientos. El error entonces se puede estimar promediando el error de validación de todos los entrenamientos. Para utilizar este método para validar un hiper-parámetro se requiere realizar una gran cantidad de entrenamientos. Por ejemplo, supongamos que se desean probar V valores diferentes de λ en un algoritmo de regresión polinómica. Por cada uno de estos valores, se deberán hacer n_{tr} entrenamientos, realizando en total $V \cdot n_{\text{tr}}$ repeticiones (en contraste con definir un conjunto de validación que solamente necesita V entrenamientos). Finalmente se elige el λ que genere menor error de validación.

Una solución intermedia entre LOOCV y definir un conjunto de validación es el método conocido como **K-Folds**. En él, se propone separar el conjunto de datos de entrenamiento en K paquetes para entrenar con $K-1$ de ellos y validar con el restante. Se repite el procedimiento K veces usando como conjunto de validación siempre un paquete diferente,

y se define el error de validación total como el promedio del error de validación de cada experimento. De esta manera, para probar V valores de un hiper-parámetro se deben realizar $V \cdot K$ entrenamientos, donde $1 < K \leq n_{\text{tr}}$ (LOOCV coincide con $K = n_{\text{tr}}$). Un valor grande de K requiere efectuar muchos entrenamientos, pero un valor chico de K reduce demasiado la cantidad de muestras efectivas de entrenamiento. Si cada paquete contiene $\frac{n_{\text{tr}}}{K}$ muestras, el entrenamiento efectivo se hará con $\frac{(K-1)\cdot n_{\text{tr}}}{K}$ muestras. La decisión de K dependerá mucho de la cantidad de muestras con las que se cuente.

3

Clasificación en Inteligencia Artificial

Utilizar un algoritmo para resolver una tarea es importante. Pero imaginemos el potencial de abrir el algoritmo y entender los por menores de su razonamiento. Quizás mirando el algoritmo nos veamos a nosotros mismos.

En la Sección 2.2.1 se discutió la problemática de no tener una relación de orden clara en predictores categóricos. Es de esperar entonces que esta problemática también sea atendida cuando la variable categórica es la etiqueta. En estos casos, el *error cuadrático medio* como métrica de error deja de tener sentido y surge la necesidad de un nuevo paradigma. Estamos en presencia del **problema de clasificación**.

Los algoritmos de clasificación son fundamentales en el mundo real. En el día a día estamos rodeados de decisiones que implican asignar categorías. Desde determinar si un correo es spam o legítimo, diagnosticar enfermedades, aprobar créditos o recomendar productos, la clasificación permite automatizar procesos que antes requerían juicio humano. En esencia, clasificar es simplificar: transformar un conjunto complejo de datos en una decisión concreta. Esta capacidad es clave para la eficiencia operativa, la personalización de servicios y la toma de decisiones informadas a gran escala. Sin algoritmos de clasificación, gran parte de las aplicaciones modernas de la inteligencia artificial simplemente no serían viables.

El objetivo de un clasificador es estimar una función $Y = \varphi(X)$, donde $Y \in \mathcal{Y}$ es una variables categórica ($\mathcal{Y}$ finito) que recibe el nombre de **clase**. La función costo elegida en este tipo de problemas es la *hard error* o *loss 0-1*: $\ell(x, y) = \mathbf{1}\{y \neq \varphi(x)\}$. Esta métrica, a diferencia del error cuadrático, solo distingue entre aciertos y errores. El riesgo esperado es la probabilidad de error $\mathbb{E}\left[\ell(X, Y)\right] = \mathbb{P}(Y \neq \varphi(X))$. Así mismo se denomina **accuracy**[1] o *probabilidad de acierto* a su probabilidad complementaria $\mathbb{P}(Y = \varphi(X))$.

Al tomar una cantidad finita de valores posibles, cada clasificador $\varphi : \mathbb{R}^{d_x} \to \mathcal{Y}$ define una partición del espacio $\mathbb{R}^{d_x}$ donde la cantidad de elementos de la partición corresponde a la cantidad de clases o elementos de $\mathcal{Y}$. A las fronteras que separan los elementos de dicha partición se las denomina **frontera de decisión**. Es importante destacar que los

[1]Se recomienda evitar traducciones literales como exactitud, ya que en el aprendizaje estadístico existen diversas métricas cuyas traducciones se pueden confundir.

elementos de la partición son *conjuntos de nivel* de la función φ.

Al igual que el problema de regresión, lo esencial para entender el objetivo de un algoritmo, es interpretar cuál es su *solución óptima* y cual es el *error bayesiano* asociado.

3.1. Optimalidad en Clasificación

En la Prop. 1.8 se demostró que el regresor asociado a la esperanza condicional minimiza el error cuadrático medio y que su error bayesiano asociado es la esperanza de la varianza condicional. En los problemas de clasificación, la función costo $\ell(x, y) = \mathbf{1}\{y \neq \varphi(x)\}$ tiene una naturaleza muy distinta al error cuadrático. Esta característica vuelve a la clasificación un poco más compleja que la regresión, y es necesario desarrollar algunos conceptos para su estudio.

3.1.1. Clasificador Bayesiano

Se denomina **clasificador bayesiano** al clasificador óptimo que minimiza la *probabilidad de error* alcanzando el *error bayesiano* [3, Capítulo 2]. Este clasificador será presentado en el siguiente resultado.

Propiedades 3.1 $\mathbb{P}\left(Y \neq \varphi(X)\right) \geq 1 - \mathbb{E}\left[\max_y P_{Y|X=X}(y)\right]$ *con igualdad si y solo si* $\varphi(x) = \arg\max_y P_{Y|X=x}(y)$.

La notación $P_{Y|X=X}(y)$ representa la probabilidad condicional $P_{Y|X=x}(y)$ evaluada en la variable aleatoria X.

Demostración 3.1 (Prop. 3.1) *La Prop. 1.7 permite definir una probabilidad como* $\mathbb{P}(Y = \varphi(X)) = \mathbb{E}[\mathbb{P}(Y = \varphi(X)|X)]$. *La probabilidad condicional dentro de la esperanza es muy fácil de acotar como:*

$$\mathbb{P}(Y = \varphi(X)|X = x) = P_{Y|X=x}(\varphi(x)) \leq \max_{y \in \mathcal{Y}} P_{Y|X=x}(y) \tag{3.1}$$

con igualdad si y solo si $\varphi(x) = \arg\max_{y \in \mathcal{Y}} P_{Y|X=x}(y)$. *Luego*

$$\mathbb{P}(Y \neq \varphi(X)) = 1 - \mathbb{E}[\mathbb{P}(Y = \varphi(X)|X)] \geq 1 - \mathbb{E}\left[\max_{y \in \mathcal{Y}} P_{Y|X=X}(y)\right] \tag{3.2}$$

Este resultado define el clasificador bayesiano como $\varphi(x) = \arg\max_y P_{Y|X=x}(y)$. El mismo no es para nada sorprendente, indica que el mejor clasificador es el que elige siempre la opción más probable. Sin embargo, el *error bayesiano* requiere un esfuerzo de interpretación. Sea $\mathcal{R}_y$, el conjunto de $x \in \mathbb{R}^{d_x}$ donde y es el máximo de $P_{Y|X=x}(y)$[2]. Es

[2]Para los $x \in \mathbb{R}^{d_x}$ donde haya dos o más máximos de $P_{Y|X=x}(y)$, se asigna dicho x a solo una de dichas $\mathcal{R}_y$ elegida arbitrariamente (en última instancia, definen conjuntos de medida nula).

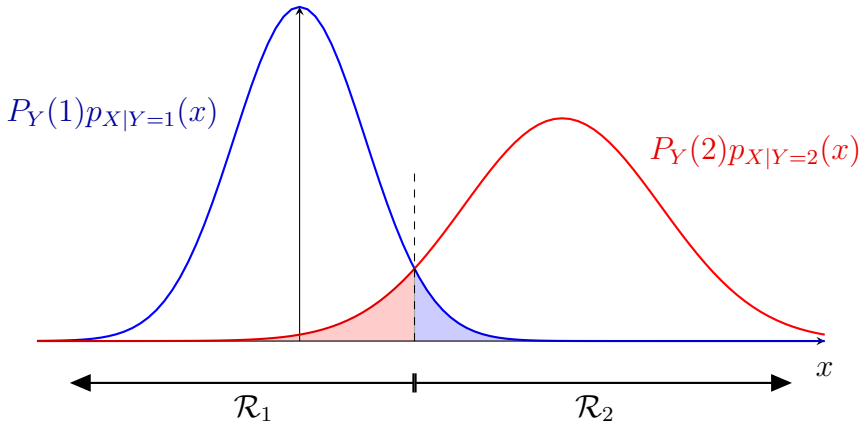


Figura 3.1: Grafico $P_Y(y)p_{X|Y=y}(x)$ para un ejemplo unidimensional de dos clases, resaltando la partición $\mathcal{R}_1$-$\mathcal{R}_2$ y el error bayesiano (sombreado).

decir que los $\mathcal{R}_y$ son los conjuntos de nivel del clasificador bayesiano:

$$\mathcal{R}_y = \left\{ x \in \mathbb{R}^{d_x} : \; P_{Y|X=x}(y) = \max_{y' \in \mathcal{Y}} P_{Y|X=x}(y') \right\} \tag{3.3}$$

Estos conjuntos definen una partición de $\mathbb{R}^{d_x}$, donde cada x pertenecer solo a uno de los posibles $\mathcal{R}_y$. Dicha partición permite reescribir el máximo de manera más amena:

$$\max_{y \in \mathcal{Y}} P_{Y|X=x}(y) = \sum_{y \in \mathcal{Y}} P_{Y|X=x}(y) \mathbf{1}\{x \in \mathcal{R}_y\} \tag{3.4}$$

$$= \sum_{y \in \mathcal{Y}} P_Y(y) \frac{p_{X|Y=y}(x)}{p_X(x)} \mathbf{1}\{x \in \mathcal{R}_y\} \tag{3.5}$$

Esta identidad reescribe el error bayesiano con una expresión fácilmente comprensible

$$1 - \mathbb{E}\left[\max_y P_{Y|X}(y|X) \right] = 1 - \int_{\mathbb{R}^{d_x}} p_X(x) \max_y P_{Y|X=x}(y) dx \tag{3.6}$$

$$= 1 - \sum_{y \in \mathcal{Y}} \int_{\mathcal{R}_y} P_Y(y) p_{X|Y=y}(x) dx \tag{3.7}$$

$$= \sum_{y \in \mathcal{Y}} P_Y(y) \mathbb{P}(X \notin \mathcal{R}_y | Y = y) \tag{3.8}$$

El resultado (3.8) expresa al error bayesiano como la suma de los errores dentro de cada región $\mathcal{R}_y$. La Fig. 3.1 muestra el producto $P_Y(y)p_{X|Y=y}(x)$ para un ejemplo unidimensional de dos clases. Las regiones $\mathcal{R}_1$ y $\mathcal{R}_2$ son delimitadas a partir de $P_{Y|X=x}(1) \lessgtr P_{Y|X=x}(2)$. Para cada $x \in \mathbb{R}$, esas regiones son equivalentes a comparar $P_Y(1)p_{X|Y=1}(x) \lessgtr P_Y(2)p_{X|Y=2}(x)$, ya que $p_X(x)$ es la misma de ambos lados[3]. El error bayesiano puede verse como la región sombreada; es la suma del área bajo la curva de $y = 1$ que cae en $\mathcal{R}_2$ (sombreado azul) y la curva de $y = 2$ que cae en $\mathcal{R}_1$ (sombreado rojo).

[3]En general, el clasificador bayesiano al definirse a partir del operador *argmax* puede escribirse como $\arg\max_y P_{Y|X=x}(y) = \arg\max_y P_Y(y)p_{X|Y=y}$.

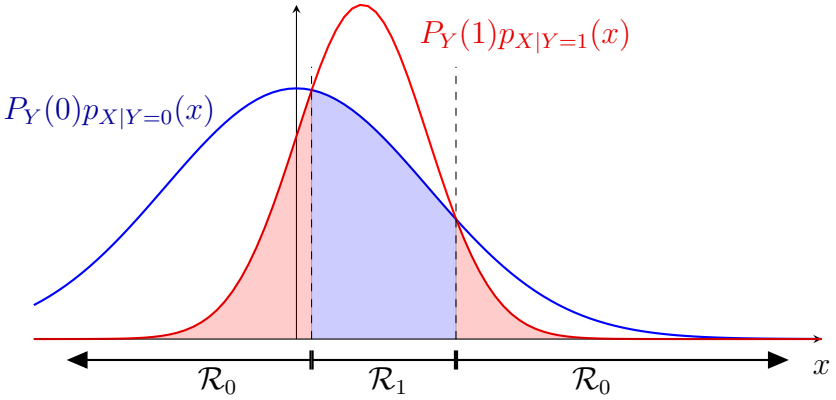


Figura 3.2: Grafico $P_Y(y)p_{X|Y=y}(x)$ para el Ej. 3.1, resaltando la partición $\mathcal{R}_0$-$\mathcal{R}_1$ y el error bayesiano (sombreado).

El error bayesiano es un límite fundamental, que no podrá ser mejorado independientemente de la tecnología. Cualquier otro clasificador siempre funcionará peor que el bayesiano. De igual manera, también existen límites razonables a superar por cualquier clasificador. Llamamos **clasificador azaroso** al que decide al azar entre todas las clases, independientemente de la entrada, siendo su probabilidad de acierto de $\frac{1}{K}$. A su vez, llamamos **clasificador dummy** al que decide siempre la clase más probable pero sin tener en cuenta la entrada, siendo su probabilidad de acierto $\max_y P_Y(y)$. Un clasificador cualquiera $\varphi(x)$ deberá cumplir entonces:

$$\sum_{y\in\mathcal{Y}} P_Y(y)\mathbb{P}(X \notin \mathcal{R}_y|Y=y) \leq \mathbb{P}(Y \neq \varphi(X)) \leq 1 - \frac{1}{K} \tag{3.9}$$

$$\sum_{y\in\mathcal{Y}} P_Y(y)\mathbb{P}(X \notin \mathcal{R}_y|Y=y) \leq \mathbb{P}(Y \neq \varphi(X)) \leq 1 - \max_y P_Y(y) \tag{3.10}$$

Cabe destacar que estos errores, tanto del clasificador azaroso como el *dummy*, no son fundamentales. Un clasificador puede tener errores peores que esto (imaginar el clasificador que se equivoca siempre). Pero el mismo carece de sentido (puede hasta ser conveniente hacer lo contrario a lo que el clasificador sugiere) desde la perspectiva del aprendizaje. A continuación se estudarán en detalle dos ejemplos para fijar los conceptos.

Ejemplo 3.1 *Por un canal de comunicaciones se emiten bits de forma aleatoria, siendo el* 40 % *de ellos* 1*. Dependiendo del bit transmitido, la comunicación es afectada por un ruido aditivo normal de media nula y varianzas:* 4 *si el bit es un* 0 *y* 1 *si el bit es un* 1*. Sea* X *la señal recibida y* Y *el bit emitido. Hallar el clasificador bayesiano y su respectivo error. Expresar el resultado en función de* $\Phi(\cdot)$ *(función de distribución de la normal estándar).*

En este tipo de ejercicios es recomendable primero determinar la distribución de los datos. En particular $Y \sim \text{Ber}(0.4)$ y X se define sumando al valor binario del bit Y

un ruido aditivo: $X|_{Y=0} \sim \mathcal{N}(0,4)$ y $X|_{Y=1} \sim \mathcal{N}(1,1)$. En la Fig. 3.2 pueden verse las curvas $P_Y(y)p_{X|Y=y}(x)$ para ambas clases. Para determinar las regiones $\mathcal{R}_0$ y $\mathcal{R}_1$ basta con analizar donde $P_Y(0)p_{X|Y=0}(x) \lessgtr P_Y(1)p_{X|Y=1}(x)$, la cuál se puede simplificar como:

$$\frac{0.6}{2\sqrt{2\pi}}e^{-\frac{x^2}{8}} \lessgtr \frac{0.4}{\sqrt{2\pi}}e^{-\frac{(x-1)^2}{2}} \tag{3.11}$$

$$\frac{3}{4} \lessgtr e^{\frac{x^2}{8}-\frac{(x-1)^2}{2}} \tag{3.12}$$

$$\log\left(\frac{3}{4}\right) \lessgtr \frac{x^2}{8} - \frac{x^2}{2} + x - \frac{1}{2} \tag{3.13}$$

$$\frac{3}{8}x^2 - x + \frac{1}{2} + \log\left(\frac{3}{4}\right) \lessgtr 0 \tag{3.14}$$

$$3x^2 - 8x + 4 - 8\log\left(\frac{4}{3}\right) \lessgtr 0 \tag{3.15}$$

El lado izquierdo de (3.15) es una párabola convexa de raíces reales x^- y x^+ que harán las veces de *frontera de decisión*. Dicha parábola será negativa ($\mathcal{R}_0$) para $x^- < x < x^+$, definiendo las regiones de la Fig. 3.2. Los valores de las raíces se pueden calcular resolviendo la parábola:

$$x^- = \frac{1}{6}\left(8 - \sqrt{16 + 96\log\left(\frac{4}{3}\right)}\right) \approx 0.23, \quad x^+ = \frac{1}{6}\left(8 + \sqrt{16 + 96\log\left(\frac{4}{3}\right)}\right) \approx 2.43 \tag{3.16}$$

El clasificador bayesiano se define a partir de que curva de la Fig. 3.2 es la más grande en cada región:

$$\varphi(x) = \arg\max_{y\in\mathcal{Y}} P_{Y|X=x}(y) = \begin{cases} 0 & x \le x^- \\ 1 & x^- < x \le x^+ \\ 0 & x^+ < x \end{cases} \tag{3.17}$$

Por último, para el error bayesiano basta con sumar los errores sombreados en la Fig. 3.2 y expresarlos a partir de la función de distribución de la normal estándar:

$$1-\mathbb{E}\left[\max_y P_{Y|X=X}(y)\right] = P_Y(0)\mathbb{P}(X \in \mathcal{R}_1|Y=0) + P_Y(1)\mathbb{P}(X \in \mathcal{R}_0|Y=1) \tag{3.18}$$

$$= 0.6\mathbb{P}(x^- < X \le x^+|Y=0) + 0.4\mathbb{P}(X \le x^-|Y=1) + 0.4\mathbb{P}(X > x^+|Y=1) \tag{3.19}$$

$$= 0.6\left[\Phi\left(\frac{x^+}{2}\right) - \Phi\left(\frac{x^-}{2}\right)\right] + 0.4\left[\Phi\left(x^- - 1\right) + 1 - \Phi\left(x^+ - 1\right)\right] \approx 0.324 \tag{3.20}$$

Dado que $Y \sim \text{Ber}(0.4)$, el error bayesiano es menor que el *dummy* $0.324 < 0.4$. Cualquier clasificador razonable tendrá una probabilidad de error entre 0.324 y 0.4.

Ejemplo 3.2 *Se quiere clasificar un conjunto de datos en dos clases. La etiqueta posee distribución $Y \sim Ber(1/2)$, mientras que los predictores $X \in \mathbb{R}^2$ poseen distri-*

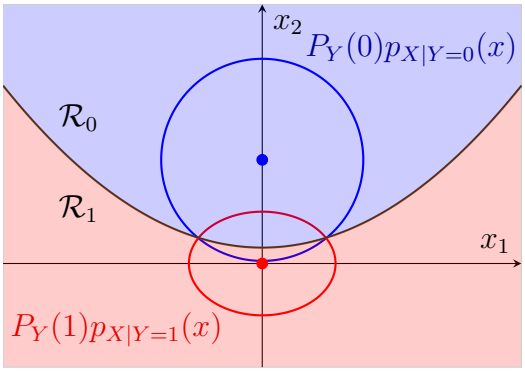


Figura 3.3: Curvas de nivel de $P_Y(y)p_{X|Y=y}(x)$ y frontera de decisión para el Ej. 3.2.

bución $X|_{Y=0} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)$ *y* $X|_{Y=1} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}\right)$. *Hallar la frontera de decisión y el clasificador bayesiano.*

En este caso se trata de un problema bidimensional. El clasificador bayesiano puede factorizarse como $\varphi(x) = \arg\max_y P_{Y|X=x}(y) = \arg\max_y P_Y(y)p_{X|Y=y}(x)$ y por lo tanto la frontera de decisión debe cumplir estará formada por todos los $x \in \mathbb{R}^2$ tales que:

$$P_Y(0)p(x|Y=0) = P_Y(1)p(x|Y=1) \tag{3.21}$$

$$\frac{1}{\sqrt{|\Sigma_0|}} e^{-\frac{1}{2}(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0)} = \frac{1}{\sqrt{|\Sigma_1|}} e^{-\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)} \tag{3.22}$$

$$\log\left(\frac{|\Sigma_1|}{|\Sigma_0|}\right) = (x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0) - (x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) \tag{3.23}$$

donde (μ_k, Σ_k) son los parámetros de $X|_{Y=k}$. Sea $x = (x_1, x_2)$, se pueden calcular todos las magnitudes involucradas en la frontera

$$\frac{|\Sigma_1|}{|\Sigma_0|} = 2 \tag{3.24}$$

$$(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0) = x_1^2 + (x_2-1)^2 \tag{3.25}$$

$$(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) = \frac{x_1^2}{2} + x_2^2 \tag{3.26}$$

La condición de la frontera de decisión puede escribirse como $\log(2) = \frac{x_1^2}{2} - 2x_2 + 1$ y por lo tanto dicha frontera es:

$$\left\{ x \in \mathbb{R}^2 : \; x_2 = \frac{x_1^2}{4} + \frac{1-\log(2)}{2} \right\} \tag{3.27}$$

En la Fig. 3.3 se puede ver la frontera de decisión y una curva de nivel de cada factorización $P_Y(y)p_{X|Y=y}(x)$. En este caso, se puede observar que para definir el clasificador bayesiano basta con verificar que las medias de las gaussianas queden del lado correcto:

$$\varphi(x_1, x_2) = \arg\max_{y \in \mathcal{Y}} P_{Y|X=(x_1,x_2)}(y) = \begin{cases} 0 & x_2 > \frac{x_1^2}{4} + \frac{1-\log(2)}{2} \\ 1 & x_2 \leq \frac{x_1^2}{4} + \frac{1-\log(2)}{2} \end{cases} \tag{3.28}$$

de esta manera quedan definidas las regiones $\mathcal{R}_0$ y $\mathcal{R}_1$.

3.1.2. Entropía Cruzada

La función costo *hard error* es problemática cuando se quiere resolver la *minimización del riesgo empírico* $\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{y_i \neq \varphi(x_i)\}$ numéricamente. No solo por los posibles problemas de diferenciabilidad, sino porque su estructura "de saltos" no permite un recorrido por gradiente descendente efectivo (es difícil de aprender). Es entonces que surge la necesidad de métodos más sofisticados. Teniendo en cuenta que el objetivo de un problema de clasificación es aprender el clasificador bayesiano $\arg\max_y P_{Y|X=x}(y)$ (véase Prop. 3.1), en la práctica suele dividirse dicha tarea en dos operaciones:

- Aprender toda la distribución condicional $P_{Y|X}(y|x)$.
- Estimar a partir de la distribución aprendida, el clasificador bayesiano quedándose con el argumento máximo $\varphi(x) = \arg\max_y P_{Y|X=x}(y)$.

Mientras que la segunda operación es inmediata, el aprender $P_{Y|X}(y|x)$ requiere definir una nueva función costo, donde minimizar su valor esperado tenga a esta distribución como valor óptimo.

Propiedades 3.2 $\mathbb{E}\left[-\log \hat{P}(Y|X)\right] \geq \mathbb{E}\left[-\log P_{Y|X=X}(Y)\right]$ *con igualdad si y solo si* $\hat{P}(y|x) = P_{Y|X}(y|x)$ *para todo* (x, y)*, donde las esperanzas involucradas son con respecto a la verdadera distribución conjunta de* (X, Y).

La demostración será presentada en la siguiente sección. La función costo **log-loss** $\ell(x, y) = -\log \hat{P}(y|x)$ puede ser estudiada de forma análoga a las Props. 1.8 y 3.1, teniendo en cuenta que se estará optimizando una relación probabilística $\hat{P}(y|x)$ en lugar de una determinística $\varphi(x)$. En este caso, el riesgo a minimizar $\mathbb{E}\left[-\log \hat{P}(Y|X)\right]$ se denomine **entropía cruzada**, el valor óptimo se alcanza con la verdadera distribución $P_{Y|X}$ y el error bayesiano asociado $\mathbb{E}\left[-\log P_{Y|X=X}(Y)\right]$ recibe el nombre de **entropía condicional**. Los clasificadores entrenados con este paradigma permiten efectuar dos tipos de decisiones:

- Sea $x \in \mathbb{R}^{d_x}$, llamamos predicción *soft* de un algoritmo a la predicción de las probabilidades estimadas $\hat{P}(\cdot|x)$. Esta estimación es un vector de probabilidades de todas las clases posibles (son valores no negativos que suman 1). Su desempeño se suele medir con el riesgo a minimizar $\mathbb{E}[-\log \hat{P}(Y|X)]$.

- Sea $x \in \mathbb{R}^{d_x}$, llamamos predicción *hard* de un algoritmo a la predicción final de la clase estimada $\varphi(x)$. Es decir, es una estimación del valor de Y. Generalmente se la suele definir a partir de la predicción *soft* como: $\varphi(x) = \arg\max_{y \in \mathcal{Y}} \hat{P}(y|x)$. Su desempeño se suele medir con la probabilidad de acierto $\mathbb{P}(Y = \varphi(X))$.

3.1.2.1. Elementos de Teoría de Información

Con el fin de interpretar las magnitudes involucradas en la Prop. 3.2 y efectuar su demostración, es necesario presentar métricas de **Teoría de Información** [2, Capítulo 2 y 8] conocidas como **entropías**. La primera métrica en cuestión busca medir la discrepancia entre distribuciones de probabilidad. Es este contexto se presenta la **divergencia de Kullback Leibler**, también conocida como **entropía relativa**.

Definición 3.1 *Sean $P(\cdot)$ y $Q(\cdot)$ dos funciones de masa de probabilidad tales que si $Q(y) = 0$ entonces $P(y) = 0$. Se define la divergencia de Kullback Leibler como:*

$$KL(P\|Q) = \sum_{y \in \mathcal{Y}} P(y) \log\left(\frac{P(y)}{Q(y)}\right) = \mathbb{E}_P\left[\log\left(\frac{P(Y)}{Q(Y)}\right)\right] \tag{3.29}$$

donde el subíndice P en la esperanza hace referencia a que medida se utiliza para calcularla.

La divergencia de *Kullback Leibler* no es una distancia ya que no es simétrica[4]. Esta característica se puede ver en el siguiente ejemplo.

Ejemplo 3.3 *Sean P y Q dos distribuciones Bernoulli de parámetros $\frac{1}{2}$ y $\frac{1}{3}$ respectivamente. Calcular $KL(P\|Q)$ y $KL(Q\|P)$.*

Las variables Bernoulli son variables de dos átomos $\{0, 1\}$ por lo que su cálculo es inmediato.

$$\mathrm{KL}\,(P\|Q) = \left(1 - \frac{1}{2}\right) \log\left(\frac{1 - \frac{1}{2}}{1 - \frac{1}{3}}\right) + \frac{1}{2} \log\left(\frac{\frac{1}{2}}{\frac{1}{3}}\right) = \frac{1}{2} \log\left(\frac{9}{8}\right) \approx 0.059 \text{ nats} \tag{3.30}$$

$$\mathrm{KL}\,(Q\|P) = \left(1 - \frac{1}{3}\right) \log\left(\frac{1 - \frac{1}{3}}{1 - \frac{1}{2}}\right) + \frac{1}{3} \log\left(\frac{\frac{1}{3}}{\frac{1}{2}}\right) = \frac{1}{3} \log\left(\frac{32}{27}\right) \approx 0.057 \text{ nats} \tag{3.31}$$

donde nats es la unidad correspondiente al logaritmo natural.

A pesar de no ser una distancia, la divergencia de *Kullback Leibler* tiene sentido como métrica discrepancia entre distribuciones debido a la siguiente propiedad.

Propiedades 3.3 *$KL(P\|Q) \geq 0$ con igualdad si y solo si $P(y) = Q(y)$ para todo $y \in \mathcal{Y}$.*

[4]Se asume la convención $0 \log(0) = 0$ por su continuidad en el límite.

Demostración 3.2 (Prop. 3.3) *Sea* $f(x) = x - 1 - \log(x)$ *con* $x > 0$*, es inmediato notar que* $f'(x) = 1 - \frac{1}{x}$ *y que* $f''(x) = \frac{1}{x^2} > 0$*. Debido a la convexidad de* $f(x)$*, la función alcanza su mínimo cuando* $f'(x) = 0$ *(se alcanza en* $x = 1$*). Por lo tanto la función* $f(x) \geq f(1) = 0$ *es no negativa. Esto quiere decir que* $\log(x) \leq x - 1$ *con igualdad si y solo si* $x = 1$*. Luego*

$$-KL\left(P\|Q\right) = \mathbb{E}_P\left[\log\left(\frac{Q(Y)}{P(Y)}\right)\right] \leq \mathbb{E}_P\left[\frac{Q(Y)}{P(Y)} - 1\right] = \mathbb{E}_P\left[\frac{Q(Y)}{P(Y)}\right] - 1 \tag{3.32}$$

con igualdad si y solo si $P(y) = Q(y)$ *para todo* $y \in \mathcal{Y}$*. Esa última esperanza puede escribirse como:*

$$\mathbb{E}_P\left[\frac{Q(Y)}{P(Y)}\right] = \sum_{y\in\mathcal{Y}} P(y)\frac{Q(y)}{P(y)} = 1 \tag{3.33}$$

Finalmente $KL(P\|Q) \geq 0$ *con igualdad si y solo si* $P(y) = Q(y)$ *para todo* $y \in \mathcal{Y}$*.*

Esta propiedad implica que la divergencia alcanza su mínimo cuando las distribuciones son iguales. Esta métrica permite proponer modelos cuyo valor óptimo sea la verdadera distribución. Con esto puede demostrarse la Prop. 3.2.

Demostración 3.3 (Prop. 3.2) *La demostración es inmediata a partir de la Prop. 3.3:*

$$\mathbb{E}\left[-\log\hat{P}(Y|X)\right] = \mathbb{E}\left[\log\left(\frac{P_{Y|X=X}(Y)}{\hat{P}(Y|X)}\right)\right] + \mathbb{E}\left[-\log P_{Y|X=X}(Y)\right] \tag{3.34}$$

$$= \mathbb{E}\left[KL\left(P_{Y|X=X}\|\hat{P}(\cdot|X)\right)\right] + \mathbb{E}\left[-\log P_{Y|X=X}(Y)\right] \tag{3.35}$$

$$\geq \mathbb{E}\left[-\log P_{Y|X=X}(Y)\right] \tag{3.36}$$

donde tener en cuenta que $KL\left(P_{Y|X=x}\|\hat{P}(\cdot|x)\right)$ *es función de* x*.*

Para terminar de interpretar la Prop. 3.2 es necesario estudiar las diferentes *entropías*. El concepto de entropía recibe su nombre por su analogía con la termodinámica y representa el valor esperado de la información que aporta conocer a una variable aleatoria [17]. Esta información depende de la probabilidad de ocurrencia de la variable a conocer. El saber que mañana saldrá el sol no contiene demasiada información porque es algo que ocurre siempre. En cambio, el saber que mañana explota el sol contiene muchísima información. En ese sentido la información es decreciente con la probabilidad. En ese sentido se definen la **entropía** y la **entropía condicional**.

$$H(Y) = \mathbb{E}[-\log P_Y(Y)], \qquad H(Y|X) = \mathbb{E}[-\log P_{Y|X=X}(Y)] \tag{3.37}$$

donde la esperanza presente en la entropía condicional es medida con respecto a la distribución conjunta de (X, Y). En este sentido $H(Y|X) = \mathbb{E}[H(P_{Y|X=X})]$, donde $H(P_{Y|X=x})$ es función de x. Además, la entropía se define mediante logaritmos porque permite transcribir en sumas y restas las factorizaciones entre probabilidades como las de (1.5) en

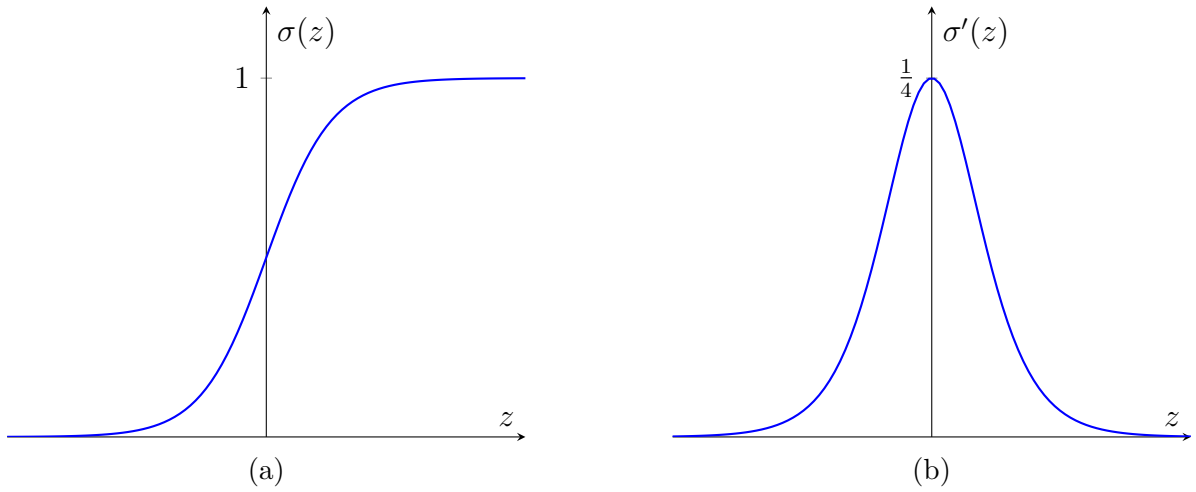


Figura 3.4: Comportamiento de (a) la función sigmoide, y (b) su derivada.

$H(X,Y) = H(Y|X) + H(X)$ (la información de (X,Y) es la suma de la información de X y la de Y que no está en X).

Por otro lado, se denomina **entropía cruzada** a las magnitudes similares a la entropía, donde la medida con la cuál se toma la esperanza difiere de la distribución dentro del logaritmo. En el caso $\mathbb{E}\left[-\log \hat{P}(Y|X)\right]$ es una *entropía cruzada* entre la verdadera $P_{Y|X=x}$ y la modelada $\hat{P}(\cdot|x)$. La entropía cruzada puede escribirse como la entropía (en este caso condicional) sumada a la divergencia de Kullback Leibler, como se mostró en (3.35). Se suele interpretar como la verdadera entropía sumada a un término de error (medida en términos de Kullback Leibler).

3.2. Regresión Logística

La regresión logística busca extender el concepto de regresión lineal estudiado en la Sección 2.2 al problema de clasificación. El objetivo es plantear una solución de muy baja complejidad (lineal), para tener ciertas garantías de que el *overfitting* está controlado. La primera diferencia es que la magnitud a estimar $\{P_{Y|X=x}(y)\}_{y\in\mathcal{Y}}$, a diferencia de $\mathbb{E}[Y|X = x]$, debe pensarse como una función vectorial (se deben estimar una función por cada átomo $\mathcal{Y}$ en lugar de solo una). Otra diferencia es que la solución debe cumplir ciertas condiciones relacionadas con ser una función de masa de probabilidad en y. Al ser la probabilidad una magnitud acotada, no será posible modelarla de forma lineal. En este tipo de problemas, lo que se busca que sea lineal es la *frontera de decisión*.

3.2.1. Regresión Logística Binaria

El caso más simple de la clasificación es la de dos clases: Sea que Y es una variable de Bernoulli[5] con $\mathcal{Y} = \{0, 1\}$. Luego es suficiente con estimar $p(x) = P_{Y|X=x}(1)$, ya que $P_{Y|X=x}(0) = 1-p(x)$ se puede calcular por el complemento (nuevamente basta con estimar un solo valor). Para garantizar que dicha estimación se encuentre en el intervalo $(0, 1)$ se modela $\hat{p}(x) = \sigma(w^T x + b)$ con $w \in \mathbb{R}^{d_x}$, $b \in \mathbb{R}$ y $\sigma(z) = \frac{1}{1+e^{-z}}$ la **función sigmoide**. Esta función cumple las siguientes propiedades.

Propiedades 3.4 *Sea $p = \sigma(z)$ la función sigmoide.*

1. *$\sigma : \mathbb{R} \to (0, 1)$ es una función creciente.*
2. *Su inversa es $\sigma^{-1}(p) = \log \frac{p}{1-p}$ con $p \in (0, 1)$.*
3. *Se derivada $\sigma'(z) = \sigma(z)(1 - \sigma(z))$ alcanza valores en $\left(0, \frac{1}{4}\right)$.*

Demostración 3.4 (Prop. 3.4) *El primer inciso es inmediato utilizando la monotonía de la exponencial y evaluando en los límites $\lim_{z\to-\infty} \sigma(z) = 0$ y $\lim_{z\to+\infty} \sigma(z) = 1$. La Fig. 3.4a muestra este comportamiento. Para el segundo inciso basta con despejar z de $p = \frac{1}{1+e^{-z}}$. Para el tercero se puede aplicar la* regla de la cadena*:*

$$\sigma'(z) = -\left(\frac{1}{1+e^{-z}}\right)^2 (-e^{-z}) = \frac{1}{1+e^{-z}} \frac{e^{-z}}{1+e^{-z}} = \sigma(z)(1-\sigma(z)) \tag{3.38}$$

Finalmente reparametrizando $\sigma'(z) = p(1-p)$, se obtiene una parábola cóncava de raíces $p = 0$ y $p = 1$. De esta manera su mínimo valor 0 se alcanza en $p = 0$ $(z \to -\infty)$ y $p = 1$ $(z \to +\infty)$, y su máximo valor $\frac{1}{4}$ se alcanza en $p = \frac{1}{2}$ $(z = 0)$. La Fig. 3.4b muestra este comportamiento.

Para analizar la forntera de decisión que una predicción *hard* genera en este tipo de modelos, se debe notar que las siguientes representaciones son equivalentes:

- $\hat{P}(1|x) = \hat{P}(0|x)$
- $\sigma(w^T x + b) = 1 - \sigma(w^T x + b)$
- $\sigma(w^T x + b) = \frac{1}{2}$
- $w^T x + b = 0$

Finalmente la frontera de decisión es lineal ya que está formada por los predictores $x \in \mathbb{R}^{d_x}$ tales que $w^T x + b = 0$. Además, las muestras x con $w^T x + b > 0$ serán clasificadas como 1, mientras que las x tales que $w^T x + b < 0$ serán clasificadas como 0.

[5] El análisis no cambia al suponer dos átomos cualesquiera.

En cambio, durante el entrenamiento se define como función costo a la *log-loss* (cuyo valor esperado era la entropía cruzada) $\ell(x, y) = -\log \hat{P}(y|x)$. Es decir que dicha función costo es de la forma:

$$\ell(x,y) = \begin{cases} -\log\big(1-\sigma(w^Tx+b)\big) & \text{Si } y=0 \\ -\log\big(\sigma(w^Tx+b)\big) & \text{Si } y=1 \end{cases} \tag{3.39}$$

$$= -(1-y)\log\big(1-\sigma(w^Tx+b)\big) - y\log\big(\sigma(w^Tx+b)\big) \tag{3.40}$$

y sus derivadas pueden calcularse a partir de la *regla de la cadena*:

$$\frac{\partial \ell(x,y)}{\partial w} = x\sigma(w^Tx+b)\big(1-\sigma(w^Tx+b)\big)\left[\frac{1-y}{1-\sigma(w^Tx+b)} - \frac{y}{\sigma(w^Tx+b)}\right] \tag{3.41}$$

$$= x\left[\sigma(w^Tx+b)(1-y) - y\big(1-\sigma(w^Tx+b)\big)\right] \tag{3.42}$$

$$= x\left[\sigma(w^Tx+b) - y\right] \tag{3.43}$$

$$\frac{\partial \ell(x,y)}{\partial b} = \sigma(w^Tx+b)\big(1-\sigma(w^Tx+b)\big)\left[\frac{1-y}{1-\sigma(w^Tx+b)} - \frac{y}{\sigma(w^Tx+b)}\right] \tag{3.44}$$

$$= \left[\sigma(w^Tx+b) - y\right] \tag{3.45}$$

En este caso, el riesgo empírico $J(w,b) = \frac{1}{n_{\text{tr}}}\sum_{i=1}^{n_{\text{tr}}} \ell(x_i, y_i)$ no tendrá un mínimo valor capaz de hallarse analíticamente, pero si por *gradiente descendente*. El gradiente en cuestión tendrá como componentes a

$$\frac{\partial J(w,b)}{\partial w} = \frac{1}{n_{\text{tr}}}\sum_{i=1}^{n_{\text{tr}}} x_i\left[\sigma(w^Tx_i+b) - y_i\right], \quad \frac{\partial J(w,b)}{\partial b} = \frac{1}{n_{\text{tr}}}\sum_{i=1}^{n_{\text{tr}}}\left[\sigma(w^Tx_i+b) - y_i\right] \tag{3.46}$$

De alguna manera, lograr que el gradientes sea nulo tiende a inducir $y_i \approx \sigma(w^Tx_i + b)$ (no tomar de forma literal, es una interpretación).

3.2.2. Regresión Logística Categórica

Adaptar el modelo de regresión logística binaria descripto en la Sección 3.2.1 a una clasificación multiclase tiene sus particularidades. Principalmente por el hecho que, por cada $x \in \mathbb{R}^{d_x}$, no se desea estimar una sola probabilidad sino una para cada $y \in \mathcal{Y}$. Sea $\mathcal{Y} = \{1, \cdots, K\}$ el conjunto de clases[6], dos de las adaptaciones más habituales se conocen como el **modelo identificable** y el **modelos softmax**.

[6]Esta decisión es solamente para simplificar la notación. Los resultados siguen siendo válidos para K átomos cualesquiera.

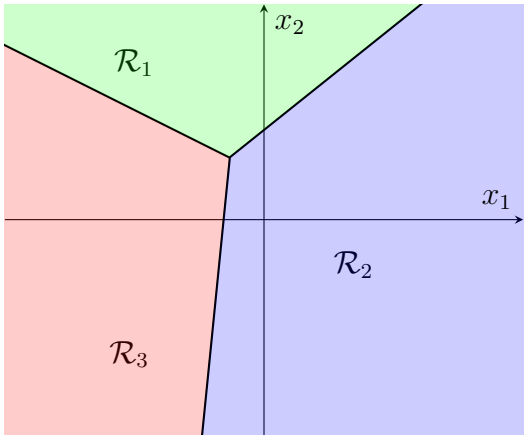


Figura 3.5: Ejemplo de frontera de decisión para un algoritmo de regresión logística categórica.

Definición 3.2 *Se denomina modelo de regresión logística identificable a*

$$\hat{P}(y|x) = \begin{cases} \frac{e^{w_y^T x + b_y}}{1+\sum_{j=1}^{K-1} e^{w_j^T x + b_j}} & y \in \{1, \cdots, K-1\} \\ \frac{1}{1+\sum_{j=1}^{K-1} e^{w_j^T x + b_j}} & y = K \end{cases} \tag{3.47}$$

donde el modelo contiene $K-1$ vectores w_j y $K-1$ valores b_j.

Definición 3.3 *Se denomina modelo de regresión logística softmax a*

$$\hat{P}(y|x) = \frac{e^{w_y^T x + b_y}}{\sum_{j=1}^{K} e^{w_j^T x + b_j}} \cdot \mathbf{1}\{y \in \{1, \cdots, K\}\} \tag{3.48}$$

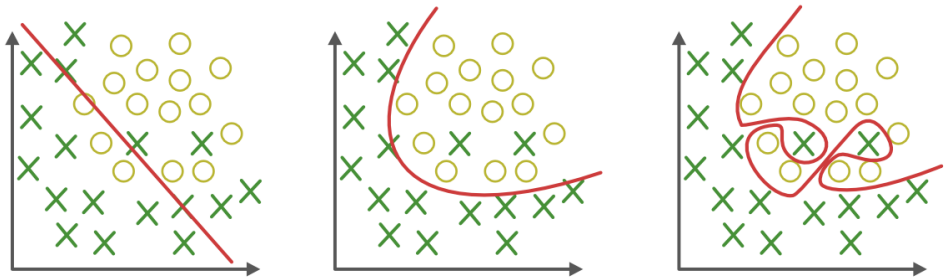
donde el modelo contiene K vectores w_j y K valores b_j.

Mientras que el primero tiene la ventaja de ser *identificable* (dos conjuntos de parámetros distintos generan modelos diferentes) y contener menos parámetros (y por lo tanto menos complejidad), el segundo es más sencillo de implementar. Por este motivo el modelo *softmax* es el más utilizado en la actualidad a pesar de estar *sobreparametrizado*; en general nos referiremos a este modelo como regresión logística categórica. El general, sea $\mathbf{a} = (a_1, \cdots, a_K)$ llamamos **softmax** a la operación matemática de $\text{softmax}(k|\mathbf{a}) = \frac{e^{a_k}}{\sum_{j=1}^{K} e^{a_j}}$. En este sentido el modelo de regresión logística softmax es el softmax de la transformación afín $w_k^T x + b_k$.

La predicción *hard*, en el caso de la regresión logística categórica softmax, posee una expresión muy simple:

$$\varphi(x) = \arg\max_{y \in \mathcal{Y}} \hat{P}(y|x) = \arg\max_{y \in \mathcal{Y}} w_y^T x + b_y \tag{3.49}$$

la cuál tiene la característica de ser lineal. Es decir que las fronteras de decisión serán hiperplanos. En la Fig. 3.5 puede verse un ejemplo de frontera para este tipo de problemas.



(a) (b) (c)

Figura 3.6: Comparación entre clasificadores denotando: (a) un ejemplo con subajuste, (b) uno con ajuste razonable, y (c) uno con sobreajuste. Imagen extraída de `https://www.geeksforgeeks.org/machine-learning/underfitting-and-overfitting-in-machine-learning/`.

Es importante destacar que la predicción *hard* es el argmax de $w_k^T x + b_k$ y la predicción *soft* el softmax de $w_k^T x + b_k$. Ambas operaciones tienen la característica de ser indiferentes a constantes sumadas (es decir, que no dependan de k).

Durante el entrenamiento se define como función costo a la *log-loss* (cuyo valor esperado era la entropía cruzada) $\ell(x, y) = -\log \hat{P}(y|x)$. Es decir que dicha función costo es de la forma:

$$\ell(x, y) = \log\left(\sum_{j=1}^{K} e^{w_j^T x + b_j}\right) - (w_y^T x + b_y) \tag{3.50}$$

y sus derivadas pueden calcularse a partir de la *regla de la cadena*:

$$\frac{\partial \ell(x,y)}{\partial w_k} = x\left(\hat{P}(k|x) - \mathbf{1}\{y = k\}\right), \qquad \frac{\partial \ell(x,y)}{\partial b_k} = \hat{P}(k|x) - \mathbf{1}\{y = k\} \tag{3.51}$$

En este caso, el riesgo empírico $J(\theta) = \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} \ell(x_i, y_i)$ (con $\theta = \{(w_y, b_y)\}_{y \in \mathcal{Y}}$) no tendrá un mínimo valor capaz de hallarse analíticamente, pero si por *gradiente descendente*. El gradiente en cuestión tendrá como componentes a

$$\frac{\partial J(\theta)}{\partial w_k} = \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} x_i \left(\hat{P}(k|x_i) - \mathbf{1}\{y_i = k\}\right), \quad \frac{\partial J(\theta)}{\partial b_k} = \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} \left(\hat{P}(k|x_i) - \mathbf{1}\{y_i = k\}\right) \tag{3.52}$$

De alguna manera, lograr que el gradientes sea nulo tiende a inducir un modelo que asigne toda su probabilidad a la etiqueta correcta $\hat{P}(k|x_i) \approx \mathbf{1}\{y_i = k\}$ (no tomar de forma literal, es una interpretación).

3.2.3. Regresión Logística Polinómica y Overfitting

El *mapa polinómico* presentado en la Sección 2.4 puede aplicarse al problema de clasificación de manera inmediata. Sea cuál sea la frontera de decisión del clasificador bayesiano

que se desea aprender, la misma puede ser aproximada con polinómios tan bien como se desee (incrementando el orden). La cantidad de parámetros de un modelo de regresión logística binaria será la misma que en el problema de regresión, mientras que en el caso de una regresión logística categórica softmax habrá que multiplicarla por el número de clases K. La normalización nuevamente es indispensable en este caso por tratarse de magnitudes de diferentes unidades y entrenar por gradiente descendente.

Al igual que en el problema de regresión, el problema de la versión polinómica de clasificación es que el aumentar la complejidad del modelo puede ocasionar *overfitting*. En la Fig. 3.6 podemos ver tres ejemplos de clasificadores. El primero propone una frontera lineal, mostrando un problema de sesgo, por presentar una complejidad insuficiente. El segundo un ajuste adecuado al problema, y el tercero un claro problema de varianza generando *overfitting*.

La manera de tratar estos fenómenos es el mismo que en regresión. El problema de sesgo se detecta por riesgo empírico alto y se soluciona aumentando la complejidad del modelo. En cambio, el problema de varianza se detecta por un alto *gap de generalización* y se soluciona incorporando más muestras o regularizando. Cabe destacar que para analizar el *underfitting* se suele focalizar en la *log-loss* como función costo ya que es el objetivo de la optimización (predicción *soft*), pero para analizar el *overfitting* se suele analizar la *loss 0-1* como función costo para analizar la capacidad de generalización (predicción *hard*). Mientras que durante el entrenamiento se minimiza la entropía cruzada empírica, en validación y testeo se analiza la probabilidad de error como objetivo último.

Una de las opciones más habituales como regularizador es utilizar la penalización L2, generando un riesgo empírico regularizado para el caso binario de la forma:

$$J(w,b) = \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n} -\log \hat{P}(y_i|x_i) + \frac{\lambda}{n_{\text{tr}}} \|w\|^2 \tag{3.53}$$

y para el caso categórico:

$$J(\theta) = \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n} -\log \hat{P}(y_i|x_i) + \frac{\lambda}{n_{\text{tr}}} \sum_{j=1}^{K} \|w_j\|^2 \tag{3.54}$$

donde si se vectorizan todos los w_j como una matriz, la suma de las normas recibe el nombre de **norma Frobenius** [9].

La etapa de validación también funciona de forma análoga a la regresión. Supongamos que se desea validar λ, entonces se realizan entrenamientos para diferentes valores del hiperparámetro minimizando en cada uno (3.53) o (3.54). Luego se decide el valor de λ que alcance menor probabilidad de error medida con el *conjunto de validación*.

Un detalle a tener en cuenta en la clasificación (que no está presente en la regresión) es la **calibración** [18]. El entrenamiento habitual de clasificación consiste en estimar $\hat{P}(y|x)$ para luego quedarse con su argumento máximo $\varphi(x) = \arg\max_y \hat{P}(y|x)$. Sin embargo, la

etapa de validación consiste en ajustar $\varphi(x)$ sin tener en cuenta las predicciones *soft*. Si el análisis o aplicación del algoritmo se va a basar en la decisión *hard* no hay problema. Pero seguir considerando $\hat{P}(y|x)$ una estimación de la probabilidad condicional puede ser problemático luego de la mencionada etapa de validación.

La calibración es una etapa posterior a la validación que re-ajusta $\hat{P}(y|x)$ deteriorando poco o nada la decisión *hard* $\varphi(x)$. Suele aplicarse sobre un *conjunto de datos de calibración* (diferente de los de entrenamiento, validación y testeo). El método más sencillo se denomina **temperature scaling** y se basa en fijar un parámetro de calibración $\tau > 0$, donde:

$$\hat{P}_{\text{cal}}(y|x) = \frac{e^{\frac{1}{\tau}(w_y^T x + b_y)}}{\sum_{j=1}^{K} e^{\frac{1}{\tau}(w_j^T x + b_j)}} \cdot \mathbf{1}\{y \in \{1, \cdots, K\}\} \tag{3.55}$$

El modelo descripto en (3.55), permite ajustar τ para ajustar la decisión *soft* sin modificar la probabilidad de error: $\arg\max_y \hat{P}_{\text{cal}}(y|x) = \arg\max_y \hat{P}(y|x)$. Una de las métricas más importante para evaluar la calibración es la entropía cruzada. Se elige $\tau > 0$ que minimice la entropía cruzada medida con el conjunto de datos de calibración. Esta métrica tiene la ventaja de poder minimizarse por gradiente descendente, en lugar de necesitar diversas pruebas como es el caso de la etapa de validación. El desempeño final del modelo se evalúa a partir de la entropía cruzada medida con el conjunto de testeo.

3.3. Figuras de Mérito en Clasificación

Posterior al entrenamiento, dependiendo de que etapa de un clasificador se desee evaluar, las métricas a analizar pueden ser diferentes. Anteriormente se mencionó como para medir el desempeño de las predicciones *soft* se utiliza la entropía cruzada, pero para medir el desempeño de las predicciones *hard* se utiliza la probabilidad de error o su complemento el *accuracy*. Sin embargo, estás métricas no son las únicas relevantes.

3.3.1. Figuras de Mérito en Clasificación Binaria

En el caso de la clasificación binaria, existe dos tipos de errores posibles: falla de detección o falsos negativos (predecir $\varphi(x) = 0$ cuando $y = 1$) y falsa alarma o falsos positivos (predecir $\varphi(x) = 1$ cuando $y = 0$). Dependiendo de la aplicación, puede que uno de ellos sea mucho más grave que el otro, volviendo a la probabilidad de error una métrica poco adecuada. En estos casos se utilizan métricas llamadas **precision** y **recall**[7].

$$\text{Precision } = \mathbb{P}\left(Y = 1 | \varphi(X) = 1\right), \qquad \text{Recall } = \mathbb{P}\left(\varphi(X) = 1 | Y = 1\right) \tag{3.56}$$

[7]Dado que muchas de las métricas poseen nombres que pueden ser considerados sinónimos, es recomendable evitar utilizar traducciones.

La métrica *precision* se utiliza cuando los falsos positivos tienen consecuencias graves. Por ejemplo, diagnosticar erróneamente una enfermedad a una persona sana. La métrica *recall* se utiliza cuando cuando los falsos negativos tiene consecuencias graves. Por ejemplo, no detectar una transacción fraudulenta. Un clasificador que cree siempre detectar $\varphi(x) = 1$, tendrá una excelente *recall*, pero no necesariamente buenas *precision* y *accuracy*.

Otro fenómeno relacionado es que ocurre cuando las clases están muy desbalanceadas (uno de los valores de Y ocurre mucho más que el otro). Es muy posible que un clasificador *dummy*, que siempre elija la clase más probable, alcance un buen *accuracy*. Desde la perspectiva de la tarea a resolver, puede que en algunos contextos esta solución sea satisfactoria. Pero desde la perspectiva del aprendizaje, el algoritmo no logró identificar ninguna característica de los datos. En conjuntos de datos desbalanceados, el desempeño del aprendizaje de un algoritmo se mide con el **F1-score**, una métrica que tiene en cuenta tanto la *precision* como la *recall*:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \cdot (\text{Precision} \,//\, \text{Recall}) \tag{3.57}$$

En el siguiente ejemplo se muestra como se estiman empíricamente las mencionadas métricas.

Ejemplo 3.4 *Un clasificador es evaluado con un conjunto de testeo. El análisis informó 120 verdaderos positivos, 60 falsos negativos, 30 falsos positivos y 9790 verdaderos negativos. Calcular accuracy, precision, recall y F1-score.*

Sea Y la verdadera etiqueta y $\varphi(X)$ el clasificador aprendido, los $n = 10000$ datos del enunciado deben ser interpretarse de la siguiente manera:

- Verdaderos Positivos: $\#(Y = 1, \varphi(X) = 1) = 120$.
- Falsos Negativos: $\#(Y = 1, \varphi(X) = 0) = 60$.
- Falsos Positivos: $\#(Y = 0, \varphi(X) = 1) = 30$.
- Verdaderos Negativos: $\#(Y = 0, \varphi(X) = 0) = 9790$.

Las estimaciones empíricas de las métricas se calculan como:

$$\text{Accuracy} = \mathbb{P}(Y = \varphi(X)) \approx \frac{\#(Y = \varphi(X))}{n} = 0.991 \tag{3.58}$$

$$\text{Precision} = \mathbb{P}(Y = 1 | \varphi(X) = 1) \approx \frac{\#(Y = 1, \varphi(X) = 1)}{\#(\varphi(X) = 1)} = 0.8 \tag{3.59}$$

$$\text{Recall} = \mathbb{P}(\varphi(X) = 1 | Y = 1) \approx \frac{\#(Y = 1, \varphi(X) = 1)}{\#(Y = 1)} \approx 0.6667 \tag{3.60}$$

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \approx 0.7273 \tag{3.61}$$

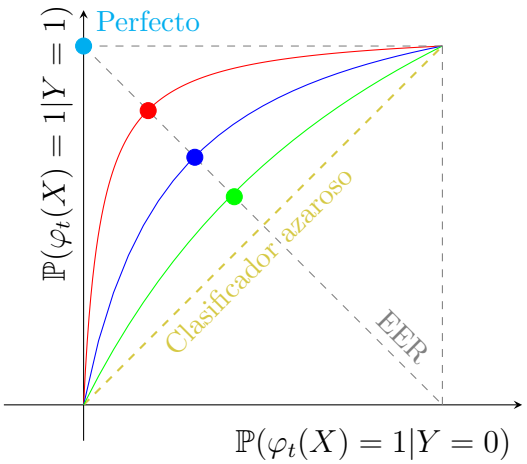


Figura 3.7: Curvas ROC para clasificadores de diferente calidad.

Mientras que el *accuracy* por si solo parece muy positivo, dependiendo de la aplicación puede que la *precision* y la *recall* sean insuficientes. El desbalance de las clases $\#(Y = 0) = 9820$ y $\#(Y = 1) = 180$ sugiere que el *accuracy* sobreestima el aprendizaje y la $F_1 \approx 0.7273$ es una métrica más representativa (es un resultado moderadamente bueno). Este fenómeno también podía detectarse comparando el *accuracy* contra la probabilidad de acierto *dummy* (siempre elegir la clase 0) 0.982. Si bien el *accuracy* es superior, su cercanía sugiere relativizar este resultado.

Una manera de corregir posibles desbalances sin repetir el entrenamiento es sesgar más a una de las clases. Por ejemplo, la clasificación binaria detecta un $y = 1$ cuando $\varphi(x) = \mathbf{1}\{\hat{P}(1|x) > \hat{P}(0|x)\}$ o su equivalente $\varphi(x) = \mathbf{1}\{\hat{P}(1|x) > \frac{1}{2}\}$. Una manera de sesgar la clasificación es comparar la probabilidad del modelo $\varphi_t(x) = \mathbf{1}\{\hat{P}(1|x) > t\}$ contra un umbral $0 \leq t \leq 1$. Para $t = 0$ el clasificador predecirá siempre $\varphi_t(x) = 1$, para $t = 1$ predecirá siempre $\varphi_t(x) = 0$, y para $t = \frac{1}{2}$ el clasificador se comportará de forma equivalente a antes de sesgar las clases $\varphi_t(x) = \varphi(x)$.

El desempeño de una familia de clasificadores $\varphi_t(x)$ se suele evaluar con las denominadas **curvas ROC** (*Receiver Operating Characteristic*). En lugar de analizar el clasificador para un umbral t, las curvas ROC permiten estudiar el desempeño de toda la familia de clasificadores posibles. Estas curvas se definen como un gráfico de la tasa de verdaderos positivos (TPR o *recall*) $\mathbb{P}(\varphi_t(X) = 1|Y = 1)$ en función de la tasa de falsos positivos (FPR) $\mathbb{P}(\varphi_t(X) = 1|Y = 0)$.

En la Fig. 3.7 pueden verse ejemplos de curvas ROC. Cada punto de la curva representa un valor de umbral distinto, pasando siempre por los puntos $(1, 1)$ (correspondiente a $t = 0$) y $(0, 0)$ (correspondiente a $t = 1$). Mientras que el clasificador perfecto se representa por el punto $(0, 1)$, suele interpretarse como límite inferior el clasificador azaroso

Figura 3.8: Ejemplo de matriz de confusión. (a) Sin normalizar; y (b) normalizando para representar $P(\hat{y}|y)$.

$\varphi_t(X) \sim \mathcal{U}(0,1)$ independiente de Y (representado por la recta identidad). En ese sentido se considerará mejor o peor algoritmo en la medida que se acerquen más o menos al óptimo o al clasificador azaroso. Por ejemplo, en la Fig. 3.7 la familia de clasificadores representada por la curva roja es mejor que la azul que a su vez es mejor que la verde.

Puede existir el caso donde que algoritmo sea mejor dependa de la región. Una métrica numérica muy utilizada para cuantificar el desempeño de una curva es el **área bajo la curva** (AUC, *Area Under the Curve*), siendo habitualmente un número entre $\frac{1}{2}$ (clasificador al azar) y 1 (clasificador perfecto). Otra métrica, un poco más sofisticada, para evaluar el desempeño de una curva ROC es el **error de igual tasa** (EER, *Equal Error Rate*). El EER se define como el error para el umbral que vuelve iguales a los dos errores $\text{EER} = 1 - \mathbb{P}(\varphi_t(X) = 1|Y = 1) = \mathbb{P}(\varphi_t(X) = 1|Y = 0)$. Básicamente está definida por la intersección entre la curva ROC con la recta $\mathbb{P}(\varphi_t(X) = 1|Y = 0) + \mathbb{P}(\varphi_t(X) = 1|Y = 1) = 1$ como puede verse en la Fig. 3.7. Esta métrica no solamente permite caracterizar toda la curva con un solo número, sino que también propone un umbral razonable para problemas donde no se conocen las consecuencias de los errores.

3.3.2. Figuras de Mérito en Clasificación Categórica

Así como la clasificación binaria tiene sus figuras de mérito particulares, la clasificación categórica también. Entre ellas, la más popular es la matriz de confusión: una matriz que representa los errores que se comente al predecir una clase en muestras de otra clase. En la Fig. 3.8a puede verse un ejemplo de matriz de confusión para una clasificación de 3 clases A, B y C. Mientras que la diagonal de la matriz muestra los aciertos, el resto de la matriz muestra errores. Por ejemplo hubo 3 muestras de la clase B, clasificadas como la clase A. Estas matrices también se puede normalizar para representar probabilidades, como puede verse en la Fig. 3.8b. Las mismas pueden normalizarse para representar $P(\hat{y}|y)$ (por fila),

$P(y|\hat{y})$ (por columna) o $P(y, \hat{y})$ (toda la matriz); siendo la primera la elegida en este ejemplo.

La métrica F1-score puede adaptarse también a este tipo de clasificación (siempre se puede adaptar, pero será recomendable utilizarla en clases desbalanceadas). Básicamente se descompone la clasificación categórica en K clasificaciones binarias del tipo *una clase contra todas*. Se denomina **Macro-F1** al promedio de los F1-score de estas clasificaciones binarias. En el ejemplo de la Fig. 3.8 se puede calcular de las siguiente manera:

- Para la clase A, se tienen 30 verdaderos positivos, 3 falsos positivos y 3 falsos negativos. Esto define una *precision*, una *recall* y un F1-score de $\frac{30}{33}$ (las tres magnitudes son iguales).
- Para la clase B, se tienen 25 verdaderos positivos, 6 falsos positivos y 5 falsos negativos. Esto define una *precision* de $\frac{25}{31}$, una *recall* de $\frac{25}{30}$ y un F1-score de $\frac{50}{61}$.
- Para la clase C, se tienen 27 verdaderos positivos, 3 falsos positivos y 4 falsos negativos. Esto define una *precision* de $\frac{27}{30}$, una *recall* de $\frac{27}{31}$ y un F1-score de $\frac{54}{61}$.

Finalmente la Macro-F1 es el promedio de las F1-score previamente calculados. En este ejemplo es de aproximandamente 0.871.

3.4. Análisis del Discriminante

En secciones anteriores se demostró que las soluciones óptimas para diferentes funciones costo se definen a partir de la distribución condicional de $Y|_{X=x}$. El error cuadrático medio se minimiza estimando $\varphi(x) = \mathbb{E}[Y|X = x]$ (Prop. 1.8), la probabilidad de error con el clasificador bayesiano $\varphi(x) = \arg\max_y P_{Y|X=x}(y)$ (Prop. 3.1) y la entropía cruzada estimando la verdadera probabilidad condicional $\hat{P}(y|x) = P_{Y|X=x}(y)$ (Prop. 3.2). En todos los casos basta con modelar la distribución condicional, sin hacer ninguna suposición sobre la marginal $p_X(x)$. Pero que sea suficiente con modelar la condicional de $Y|_{X=x}$ no impide modelar también la marginal de X.

Una primera distinción entre algoritmos de aprendizaje estadístico es la de algoritmos **discriminativos** y **generativos**. Se llama algoritmo discriminativos a los que solamente modelan la distribución condicional $Y|_{X=x}$ y generativos a los que modelan la conjunta (X, Y). Los discriminativos poseen la ventaja de imponer menos hipótesis, modelando solamente la relación que se encarga de la predicción. En contraste, los algoritmos generativos requieren modelar también la distribución marginal de los datos por lo que necesitan mayor cantidad de datos para estimarse. Pero estos últimos poseen la ventaja de poder *generar nuevos datos sintéticos* $X \sim p_X$.

Con la popularización de la inteligencia artificial, es por todos conocida la utilidad de los algoritmos generativos. Actualmente nos preguntamos como pudimos vivir tantos años sin una máquina que dibuje un conejo realista disfrazado de ángel. Las aplicaciones de los modelos generativos pueden ser relevantes por si misma (como por ejemplo los **modelos de lenguaje**) o pueden servir para mejorar la etapa discriminativa. Se denomina **aumento de datos**, al procedimiento de incorporar datos sintéticos previo a un entrenamiento. Sin embargo, estos datos deben usarse con cuidado porque no dejan de ser artificiales y de ninguna manera independientes a los datos utilizados para generarlos.

Uno de los modelos generativos más sencillo es el de mezcla de gaussianas (GMM, Gaussian Mixture Model). En él, se asume tanto que Y es una variable categórica $Y \sim \text{Cat}(c_1, \cdots, c_K)$ como que la distribución de $X|_{Y=k} \sim \mathcal{N}(\mu_k, \Sigma_k)$. Es importante resaltar que, una vez estimados los parámetros del modelo, será posible simular nuevas realizaciones como $k \leftarrow \text{Cat}(c_1, \cdots, c_K)$ para luego simular $x \leftarrow \mathcal{N}(\mu_k, \Sigma_k)$. En el GMM, la distribución de los predictores es una mezcla de gaussianas:

$$\hat{p}(x) = \sum_{k=1}^{K} \frac{c_k e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}}{(2\pi)^{\frac{d_x}{2}} \sqrt{|\Sigma_k|}} \tag{3.62}$$

y además la distribución condicional es de la forma:

$$\hat{P}(y|x) \propto e^{\log(c_y) - \frac{1}{2}(x-\mu_y)^T \Sigma_y^{-1}(x-\mu_y) - \frac{1}{2}\log|\Sigma_y|} \tag{3.63}$$

En un modelo GMM, $\hat{P}(y|x)$ es el **softmax** de la forma cuadrática $\delta_k(x) = \log(c_k) - \frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\Sigma_k|$, la cuál recibe el nombre de función discriminante. La función discriminante define las fronteras de decisión del problema[8]. El Ejemplo 3.2 es un caso de GMM, donde se observa una frontera cuadrática. Por este motivo, el algoritmo utilizado para modelar la GMM recibe el nombre de **análisis del discriminante cuadrático** (QDA, *Quadratic Discriminant Analysis*), donde las predicciones *hard* son definidas a partir de

$$\varphi(x) = \arg\max_{y \in \mathcal{Y}} \hat{P}(y|x) = \arg\max_{y \in \mathcal{Y}} \delta_y(x) \tag{3.64}$$

De esta manera, la predicción *soft* $\hat{P}(y|x)$ y la predicción *hard* $\varphi(x)$ serán el softmax y el argmax de la función discriminante $\delta_y(x)$ respectivamente.

Uno de los problemas de los modelos generativos es que al estimar tanto $P_{Y|X=x}(y)$ como $p(x)$, los modelos poseen mayor cantidad de parámetros. Esta característica vuelve a estos modelos más complejos que sus contra-partes discriminativas, aumentando los riesgos de *overfitting* si no se tiene suficiente cantidad de muestras. Con el fin de bajar la complejidad, una habitual simplificación es suponer todas las matrices de covarianza iguales ($\Sigma_y = \Sigma$ para todo $y \in \mathcal{Y}$), hipótesis que recibe el nombre de homocedasticidad. Bajo esta hipótesis, la distribución condicional puede reescribirse agrupando constantes

[8]Las fronteras pueden parametrizarse como $\delta_i(x) = \delta_j(x)$ con $i \neq j$.

multiplicativas como:

$$\hat{P}(y|x) \propto e^{\log(c_y) - \frac{1}{2}(x-\mu_y)^T \Sigma^{-1}(x-\mu_y)} \propto e^{\log(c_y) + x^T\Sigma^{-1}\mu_y - \frac{1}{2}\mu_y^T\Sigma^{-1}\mu_y} \tag{3.65}$$

donde se redefine la función discriminante para este caso como una función lineal $\tilde{\delta}_k(x) = x^T\Sigma^{-1}\mu_k + \log(c_k) - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k$. Por este motivo, el algoritmo utilizado para modelar la GMM recibe el nombre de **análisis del discriminante lineal** (LDA, *Linear Discriminant Analysis*), donde la predicción *soft* $\hat{P}(y|x)$ y la predicción *hard* $\varphi(x)$ serán el softmax y el argmax de la función discriminante $\tilde{\delta}_y(x)$ respectivamente. Es importante destacar que el modelo asociado a la regresión logística, analizada en la Sección 3.2, es un caso particular de LDA donde $w_k = \Sigma^{-1}\mu_k$ y $b = \log(c_k) - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k$.

Una de las características de los algoritmos de LDA y QDA es que ambos desarrollan su etapa de entrenamiento a partir de computar estimadores puntuales. Es habitual utilizar para ello estimadores insesgados:

$$\mathcal{D}_k = \{x_i : \ 1 \leq i \leq n \ \wedge \ y_i = k\} \tag{3.66}$$

$$\hat{c}_k = \frac{\#(\mathcal{D}_k)}{n} \tag{3.67}$$

$$\hat{\mu}_k = \frac{1}{\#(\mathcal{D}_k)} \sum_{x \in \mathcal{D}_k} x \tag{3.68}$$

$$\hat{\Sigma}_k = \frac{1}{\#(\mathcal{D}_k) - 1} \sum_{x \in \mathcal{D}_k} (x - \hat{\mu}_k)(x - \hat{\mu}_k)^T \tag{3.69}$$

$$\hat{\Sigma} = \frac{1}{n - K} \sum_{k=1}^{K} (\#(\mathcal{D}_k) - 1)\hat{\Sigma}_k \tag{3.70}$$

donde en el caso de QDA se utilizan las $\hat{\Sigma}_k$ (no es necesario calcular $\hat{\Sigma}$) y en el caso de LDA las $\hat{\Sigma}_k$ solamente son un paso intermedio en el cálculo de la covarianza del modelo $\hat{\Sigma}$. A continuación se demostrará por que este tipo de estimadores son insesgados.

Demostración 3.5 (Estimadores Insesgados) *En primer lugar supongamos que* $X|_{Y=k} \sim \mathcal{N}(\mu_k, \Sigma_k)$ *y* $Y \sim Cat(c_1, \cdots, c_K)$. *Entonces* $\#(\mathcal{D}_k) \sim Bin(n, c_k)$ *y por lo tanto* $\mathbb{E}[\hat{c}_k] = c_k$. *En segundo lugar, se puede observar que las medias pueden escribirse como* $\hat{\mu}_k = \frac{1}{\#(\mathcal{D}_k)} \sum_{i=1}^{n} x_i \mathbf{1}\{y_i = k\}$. *Se define* $\mathbf{y} = (y_1, \cdots, y_n)$, *donde* $\#(\mathcal{D}_k) = f(\mathbf{y})$ *es función de las etiquetas. Utilizando las propiedades de la esperanza condicional (Props. 1.7):*

$$\mathbb{E}[\hat{\mu}_k] = \mathbb{E}[\mathbb{E}[\hat{\mu}_k|\mathbf{y}]] = \mathbb{E}\left[\frac{1}{\#(\mathcal{D}_k)} \sum_{i=1}^{n} \mathbb{E}\left[x_i|\mathbf{y}\right] \mathbf{1}\{y_i = k\}\right] \tag{3.71}$$

Dado que los pares (x, y) *son independientes entre ellos,* $\mathbb{E}\left[x_i|\mathbf{y}\right] = \mathbb{E}\left[x_i|y_i\right] = \mu_{y_i}$. *Por lo tanto,*

$$\mathbb{E}[\hat{\mu}_k] = \mathbb{E}\left[\frac{\mu_k}{\#(\mathcal{D}_k)} \sum_{i=1}^{n} \mathbf{1}\{y_i = k\}\right] = \mu_k \tag{3.72}$$

Repitiendo la misma idea con las covarianzas se observa

$$\mathbb{E}[\hat{\Sigma}_k] = \mathbb{E}[\mathbb{E}[\hat{\Sigma}_k|\mathbf{y}]] = \mathbb{E}\left[\frac{1}{\#(\mathcal{D}_k)-1}\sum_{i=1}^{n}\mathbb{E}\left[(x_i-\hat{\mu}_k)(x_i-\hat{\mu}_k)^T|\mathbf{y}\right]\mathbf{1}\left\{y_i=k\right\}\right] \tag{3.73}$$

donde el producto dentro de la esperanza puede reescribirse como $(x_i-\hat{\mu}_k)(x_i-\hat{\mu}_k)^T = x_i x_i^T - x_i\hat{\mu}_k^T - \hat{\mu}_k x_i^T + \hat{\mu}_k\hat{\mu}_k^T$*. La esperanza de cada uno de estos términos puede escribirse como:*

$$\mathbb{E}[x_i x_i^T|\mathbf{y}] = \Sigma_{y_i} + \mu_{y_i}\mu_{y_i}^T \tag{3.74}$$

$$\mathbb{E}[x_i\hat{\mu}_k^T|\mathbf{y}] = \frac{1}{\#(\mathcal{D}_k)}\sum_{j=1}^{n}\mathbb{E}[x_i x_j^T|\mathbf{y}]\mathbf{1}\left\{y_j=k\right\} \tag{3.75}$$

$$= \frac{1}{\#(\mathcal{D}_k)}\left(\Sigma_k + \mu_k\mu_k^T + (\#(\mathcal{D}_k)-1)\mu_{y_i}\mu_k^T\right) \tag{3.76}$$

$$\mathbb{E}[\hat{\mu}_k x_i^T|\mathbf{y}] = \frac{1}{\#(\mathcal{D}_k)}\left(\Sigma_k + \mu_k\mu_k^T + (\#(\mathcal{D}_k)-1)\mu_k\mu_{y_i}^T\right) \tag{3.77}$$

$$\mathbb{E}\left[\hat{\mu}_k\hat{\mu}_k^T|\mathbf{y}\right] = \frac{1}{\#(\mathcal{D}_k)^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbb{E}[x_i x_j^T|\mathbf{y}]\mathbf{1}\left\{y_i=y_j=k\right\} \tag{3.78}$$

$$= \frac{1}{\#(\mathcal{D}_k)^2}\left(\#(\mathcal{D}_k)(\Sigma_k+\mu_k\mu_k^T) + (\#(\mathcal{D}_k)^2 - \#(\mathcal{D}_k))\mu_k\mu_k^T\right) \tag{3.79}$$

Teniendo en cuenta los signos y la indicadora, cuando se haga la suma en (3.73)*, el producto* $\mu_k\mu_k^T$ *se cancelará ya que aparecerá la misma cantidad de veces sumando que restando. En cambio, la covarianza* Σ_k *aparecerá* $\#(\mathcal{D}_k)-1$ *veces, y por lo tanto, el valor esperado de cada covarianza es* $\mathbb{E}[\hat{\Sigma}_k] = \Sigma_k$*.*

Por último, para el caso de LDA, supongamos que $X|_{Y=k} \sim \mathcal{N}(\mu_k, \Sigma)$ *y* $Y \sim Cat(c_1, \cdots, c_K)$*. En este caso, las verdaderas covarianzas son iguales pero los estimadores* $\hat{\Sigma}_k$ *son distintos. Los mismos se combinan como:*

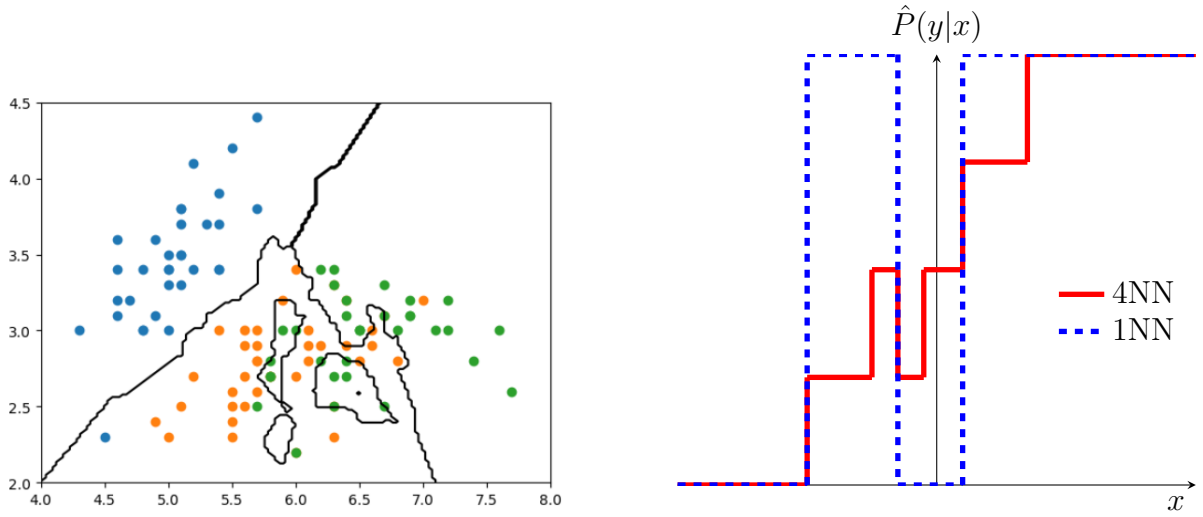
$$\mathbb{E}[\hat{\Sigma}] = \frac{1}{n-K}\sum_{k=1}^{K}\mathbb{E}\left[(\#(\mathcal{D}_k)-1)\hat{\Sigma}_k\right] \tag{3.80}$$

Repitiendo de forma análoga al procedimiento anterior para $(\#(\mathcal{D}_k)-1)\hat{\Sigma}_k$ *se finaliza la demostración:*

$$\mathbb{E}[\hat{\Sigma}] = \frac{1}{n-K}\sum_{k=1}^{K}\mathbb{E}\left[(\#(\mathcal{D}_k)-1)\right]\Sigma = \frac{\sum_{k=1}^{K}(nc_k-1)}{n-K}\Sigma = \Sigma \tag{3.81}$$

3.5. Vecinos más Cercanos

Los algoritmos descriptos hasta el momento se denominan **modelos paramétricos** por asumir una familia para $\hat{P}(y|x)$ indexada por medio de parámetros. En contraposición, se definen los **modelos no paramétricos** a los algoritmos cuyos modelos no están definidos



(a) (b)

Figura 3.9: Soluciones típicas del problema de KNN. (a) Frontera de decisión (*hard*), y (b) probabilidades estimadas (*soft*).

por parámetros. Un ejemplo de este tipo de algoritmos es el histograma presentado en la Sec. 1.2.1, el cuál estima distribuciones en general fuera del paradigma de clasificación.

La adaptación habitual del histograma a problemas de clasificación es bastante inmediata: Se modela $\hat{P}(y)$ con su distribución empírica y $\hat{p}(x|y)$ como una densidad constante por regiones (al igual que el histograma). Es decir, $\hat{P}(y) = \frac{N_y}{n_{\text{tr}}}$ donde N_y es la cantidad de muestras correspondientes a la clase y-ésima y n_{tr} la cantidad total de muestras de entrenamiento, y $\hat{p}(x|y) = \frac{K_y}{N_y \cdot V}$ donde V es el volumen de la región a la que pertenece el valor x y K_y la cantidad de muestras, etiquetadas como y, que pertenecen en dicha región. Luego la probabilidad condicional (decisión *soft*) es de la forma:

$$\hat{P}(y|x) = \frac{\hat{P}(y)\hat{p}(x|y)}{\sum_{k\in\mathcal{Y}} \hat{P}(k)\hat{p}(x|k)} = \frac{K_y}{K} \tag{3.82}$$

donde $K = \sum_{y\in\mathcal{Y}} K_y$ es la cantidad de muestras totales pertenecientes a la región en cuestión. El resultado es esperable, la probabilidad se modela como una proporción de muestras de cada clase en la región.

El algoritmo no paramétrico más sencillo se conoce como $K-$**vecinos más cercanos** (KNN, $K-$nearest neighbors) [3, Sección 4.4]. Mientras que el histograma prefija el volumen de las celdas V y computa la cantidad de muestras que caen en esa celda K, el método de KNN prefija la cantidad de *vecinos* y adapta el tamaño de las celdas. Es decir, para computar (3.82) simplemente analizar la proporción de etiquetas en los $K-$vecinos más cercanos.

Por el lado de las decisiones *hard*, la frontera de decisión en este tipo de problemas se puede definir por la clase mayoritaria dentro del vecindario $\varphi(x) = \arg\max_y \hat{P}(y|x) =$

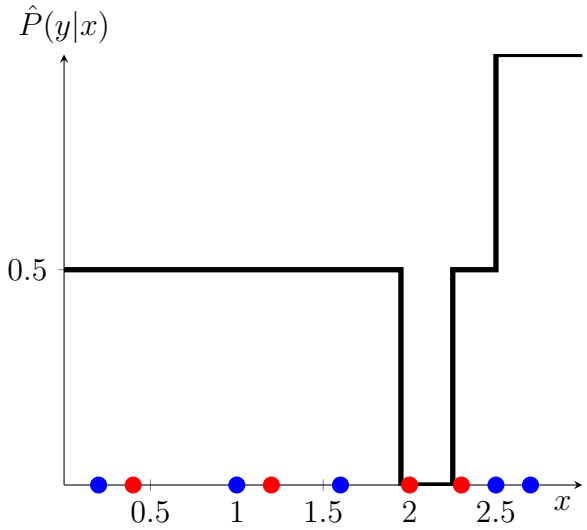


Figura 3.10: Probabilidades estimadas para el Ej. 3.5.

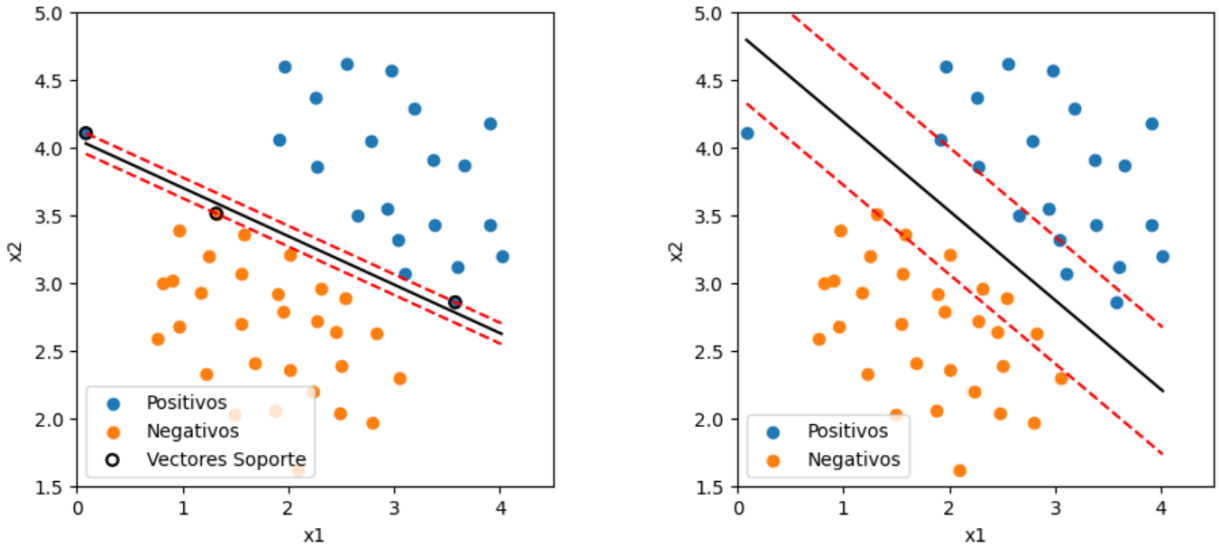
$\arg\max_y K_y$. Un ejemplo habitual de frontera de decisión puede verse en la Fig. 3.9a. En cambio, las decisiones *soft* se puede ver que las probabilidades solo podrán tomar valores posibles $\frac{k}{K}$ con $k \in \{0, \cdots, K\}$. Un ejemplo habitual de estas probabilidades estimadas puede verse en la Fig. 3.9b, donde se puede observar la diferencia entre un algoritmo 1NN y 4NN. Un ejemplo de uso de este tipo de algoritmos puede verse a continuación.

Ejemplo 3.5 *Graficar $\hat{P}(1|x)$ para una clasificación binaria mediante un algoritmo 2NN cuyos datos de entrenamiento pueden verse en la siguiente tabla.*

X	1.2	0.2	2.3	0.4	2.0	1.6	1.0	2.5	2.7
Y	0	1	0	0	0	1	1	1	1

Este tipo de ejemplos son ideales para terminar de comprender los detalles del algoritmo. Se recomienda comenzar el ejercicio ubicando las muestras sobre el eje de las abscisas como se muestra en la Fig. 3.10, en este caso en rojo las de $y = 0$ y en azul las de $y = 1$. Para comprender como se forma la estimación $\hat{P}(y|x)$ tomemos $x = 0.5$ como ejemplo. Las muestras más cercanas son $x = 0.4$ (etiquetada como $y = 0$) y $x = 0.2$ (etiquetada como $y = 1$). En este caso 1 de 2 muestras poseen etiqueta 1 y por lo tanto $\hat{P}(1|0.5) = \frac{1}{2}$.

Generalizando esta metodología a cualquier $x \in \mathbb{R}$, podemos observar que para x negativos, los dos vecinos más cercanos corresponden a etiquetas diferentes ($\hat{P}(1|0.5) = \frac{1}{2}$). El foco debe colocarse en que valores hacen modificar esta probabilidad. A medida que se va incrementando x, se alcanza un punto donde la muestra de $x = 0.2$ deja de ser uno de los dos vecinos más cercanos y lo reemplaza $x = 1$, pero junto con $x = 0.4$ siguen siendo dos



(a) (b)

Figura 3.11: Soluciones típicas al problema de SVM. (a) Versión estándar, y (b) con márgenes relajados.

vecinos de etiquetas diferentes. Este comportamiento cambia a partir de que los vecinos más cercanos sean $x = 2$ y $x = 2.3$ ambos con etiquetas $y = 0$ y por lo tanto $\hat{P}(1|0.5) = 0$. Esta tendencia se da cuando la muestra $x = 2.3$ está más cerca que la de $x = 1.6$, es decir cuando $x > \frac{1.6+2.3}{2} = 1.95$. La tendencia vuelve a cambiar cuando la muestra de $x = 2.5$ se vuelve más cercana que la de $x = 2$, lo cuál ocurre a partir de $x > 2.25$ (etiquetas diferentes). El último cambio se da cuando los dos vecinos comienzan a tener etiqueta $y = 1$, lo cuál ocurre a partir de $x > 2.5$. La probabilidad estimada puede verse en la Fig. 3.10.

3.6. Máquina de Vectores Soporte

La máquina de vectores soporte (*support vector machine* o SVM), es un algoritmo de aprendizaje supervisado con un enfoque diferente a otros métodos previamente mencionados. Su criterio de optimalidad se basa en maximizar el márgen resultante entre las muestras y la frontera de decisión.

3.6.1. SVM estándar

Supongamos un problema de clasificación binaria $y \in \{-1, 1\}$, en el cuál se propone una frontera de decisión lineal con ecuación $z(x) = w^T \cdot x + b = 0$. Esta parametrización permite clasificar una muestra a partir de $\varphi(x) = \text{SIGNO}[z(x)]$ (si bien este planteo no

tiene una estructura probabilística, se puede considerar a $z(x)$ como una *clasificación soft*).

Una muestra aleatoria se dice *linealmente separables* si existe una solución lineal que separa perfectamente las muestras de ambas clases (error de entrenamiento nulo). Matemáticamente esto se traduce en $y \cdot z(x) > 0$ para toda muestra de entrenamiento (x, y). La versión estándar de SVM busca encontrar la frontera óptima para los conjuntos de datos que poseen esta característica. En este contexto, se define $f_i(w, b) = y_i z(x_i) > 0$ para todo $1 \leq i \leq n_{\text{tr}}$. En la Fig. 3.11a puede verse un ejemplo de clasificación con el criterio de SVM para el tipo de problema descripto anteriormente.

Algunos razonamientos son necesarios para expresar el problema de maximizar el márgen en términos matemáticos. Por un lado, la ecuación $z(x) = w^T \cdot x + b = 0$ define un hiperplano cuya dirección normal es w y por lo tanto este vector es ortogonal a la frontera. Sea x_* la proyección ortogonal de x sobre la frontera, es fácil notar que w es paralelo a $(x - x_*)$ y por lo tanto:

$$|w^T(x - x_*)| = \|w\| \cdot \|x - x_*\| \tag{3.83}$$

Por otro lado, $z(x_*) = 0$ ya que x_* pertenece a la frontera y por lo tanto $z(x) = z(x) - z(x_*) = w^T(x - x_*)$. Esto nos permite expresar la distancia entre la muestra i-ésima y la frontera $\|x_i - x_*\|$ como:

$$\|x_i - x_*\| = \frac{|w^T(x_i - x_*)|}{\|w\|} = \frac{|z(x_i)|}{\|w\|} = \frac{y_i \cdot z(x_i)}{\|w\|} = \frac{f_i(w, b)}{\|w\|} \tag{3.84}$$

donde se utilizó que las muestras son linealmente separables. Se define el margen unilateral $m(w, b)$ como un criterio de peor caso para la distancia entre la muestra y la frontera:

$$m(w, b) = \min_{1 \leq i \leq n_{\text{tr}}} \frac{f_i(w, b)}{\|w\|} = \frac{f_k(w, b)}{\|w\|} \tag{3.85}$$

donde k es el índice óptimo para esa minimización (y por lo tanto será función de w y b). Un primer criterio de optimalidad de SVM consiste en:

$$\max_{w,b} \; m(w, b) \quad \text{s.t.} \quad f_i(w, b) > 0 \quad \forall \; 1 \leq i \leq n_{\text{tr}} \tag{3.86}$$

Sea $\alpha > 0$, esta claro que la regla de decisión $z(x) \gtrless 0$ (o el clasificador $\varphi(x) = \text{SIGNO}[z(x)]$) no se ve afectada si se reescalan consistentemente los parámetros $w \leftarrow \alpha w$ y $b \leftarrow \alpha b$. Esto mismo ocurre con el margen unilateral, quien es invariante a este tipo de cambio de escala $m(\alpha w, \alpha b) = m(w, b)$. Este fenómeno está denotando la característica *no identificable* del modelo (véase Sección 3.2.2): hay infinitas maneras posibles de describir una solución. SVM toma esta característica y la utiliza para simplificar el problema, resaltando que $f_k(\alpha w, \alpha b) = \alpha f_k(w, b)$. Se elegirá entonces una escala tal que $f_k(w, b) = 1$, logrando un margen de $m(w, b) = \frac{1}{\|w\|}$ y un $f_i(w, b) \geq 1$ para todo $1 \leq i \leq n$.

Las muestras donde $f_i(w,b) = 1$ ($y_i(w^T \cdot x + b) = 1$), se denominan **vectores soportes**. Tal como se puede ver en la Fig. 3.11a, estas muestras son las que tocan el margen, volviéndose un soporte para toda la regla de decisión (la cual queda definida solo por ellas). Tienen la característica de tener al menos una por clase. De esta manera, el problema (3.86) se puede reescribir (monotonía mediante) como:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_i(w^T \cdot x_i + b) \geq 1 \quad \forall\, 1 \leq i \leq n_{\text{tr}} \tag{3.87}$$

El problema (3.87) se conoce como **problema primal** de SVM estándar y es un ejemplo de un problema de optimización cuadrática. En la práctica suele implementarse una versión alternativa del problema conocida como **problema dual** por otorgar ciertas ventajas; más adelante se explorará esta equivalencia.

La versión estándar de SVM tiene algunas limitaciones: es muy sensible a outliers (por ser un criterio de peor caso), solamente permite soluciones lineales y resuelve únicamente el problema de clasificación binaria. Estas limitaciones son un inconveniente en la vida real, por lo que se suelen solucionar *emparchando* la versión estándar. Es decir, se plantean pequeñas variantes al problema con el fin de mitigar dichos inconvenientes.

3.6.2. Parches

En primer lugar, se plantea una variante donde se permita la penetración de muestras dentro del margen. Esto permite insensibilizar un poco a la frontera de decisión frente a un *outlier*. La Fig. 3.11b muestra una solución con márgenes relajado a modo de ejemplo. Esto se logar redefiniendo el problema primal (3.87) de la siguiente manera:

$$\min_{w,b,\xi_1,\cdots,\xi_{n_{\text{tr}}}} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n_{\text{tr}}} \xi_i \quad \text{s.t.} \quad \begin{cases} y_i(w^T x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \quad \forall\, 1 \leq i \leq n_{\text{tr}} \tag{3.88}$$

donde $C > 0$ es un hiperparámetro que controla la penetración de las muestras sobre el margen. El importante resaltar que este problema coincide con (3.87) cuando $C \to \infty$ (lejos de lo que uno podría pensar a simple vista). Esto ocurre porque un C muy grande obliga a los ξ_i a tender a cero (notar que los ξ_i forman parte de las variables a optimizar). En cambio, si se elige un $C \to 0$, no habrá nada que evite que $\xi_i \to \infty$, por lo que $w \to 0$.

Las variables ξ_i indican el grado de penetración de cada muestra. Las muestras que respetan los márgenes cumplen $y_i(w^T x_i + b) \geq 1$ (sin necesidad de los ξ_i), y por lo tanto la minimización llevará estas variables a cero $\xi_i = 0$. Un $0 < \xi_i < 1$ indica una muestra que penetró el margen pero no traspasó la frontera (sigue siendo clasificada correctamente), mientras un $\xi_i > 1$ indica una muestra clasificada erróneamente.

Una segunda característica a emparchar es el tema de la linealidad. No siempre este

tipo de soluciones son adecuadas, sino que además no cualquier conjunto de muestras es *linealmente separable*. Una solución a este problema es proyectar las muestras sobre un espacio de mayor dimensión (quizás hasta infinita) o complejidad $\phi(x_i)$, para luego poder separar las clases linealmente:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_i(w^T \cdot \phi(x_i) + b) \geq 1 \quad \forall\, 1 \leq i \leq n_{\text{tr}} \tag{3.89}$$

donde eligiendo adecuadamente esa transformación $\phi(\cdot)$, se puede reconfigurar el problema para que finalmente sea linealmente separable.

Como tercer parche surge la necesidad de adaptar este algoritmo a una tarea de clasificación categórica. Existen dos maneras clásicas de adaptar algoritmos de clasificación binaria a mayor cantidad de clases, ambos basados en la idea de entrenar muchos clasificadores binarios en paralelo.

- *one-vs-one*: Se entrenan clasificadores binarios, teniendo en cuenta todas las combinaciones de pares de clases. Esto hace un total de $\frac{K(K-1)}{2}$ clasificadores. La clasificación final se efectúa seleccionando a la clase con más *votos*.

- *one-vs-the-rest*: Se entrenan K clasificadores binarios, donde cada uno toma una clase como positiva y el resto como negativa. Se clasifica según $\varphi(x) = \arg\max_k w_k^T \phi(x) + b_k$.

Mientras que *one-vs-one* tiene la ventaja de soler alcanzar el mejor desempeño, por probar todas las combinaciones, tiene la desventaja de necesitar repetir mayor cantidad de veces el entrenamiento $\mathcal{O}(K^2)$. En cambio, *one-vs-the-rest*, si bien suele ser subóptimo, necesita solamente K entrenamientos.

Finalmente, como último parche, cabe mencionar que este algoritmo puede adaptarse para una tarea de regresión. El algoritmo de SVM para regresión resuelve el siguiente problema:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad |w^T \cdot x_i + b - y_i| \leq \epsilon \quad \forall\, 1 \leq i \leq n_{\text{tr}} \tag{3.90}$$

para algún $\epsilon > 0$ pequeño. La condición de clases linealmente separables se reemplaza por error absoluto de estimación lineal de a lo sumo ϵ. Es decir, dentro de todas las soluciones lineales con error de entrenamiento[9] a lo sumo ϵ, se elige la de menor $\|w\|$. Esta idea de elegir parámetros de norma baja, viene de la mano con combatir el *overfitting*, tal como se discutió en la Sec. 2.4.2.

[9]Error absoluto y por muestra (criterio de peor caso). No confundir con el error cuadrático medio.

3.6.3. Dualidad en SVM

La dualidad es una de las ideas centrales de la optimización matemática: muchos problemas pueden analizarse desde dos perspectivas complementarias, una "primal", que describe directamente la tarea a resolver, y otra "dual", que captura las limitaciones y/o costos asociados a las restricciones. Esta visión dual no solo aporta interpretación geométrica, sino que también permite obtener cotas, condiciones de optimalidad y, en numerosos casos, reformular el problema de manera más eficiente o más fácil de resolver. En problemas convexos, y bajo hipótesis adecuadas, ambas perspectivas coinciden exactamente, dando lugar a la llamada dualidad fuerte, fundamento teórico de numerosos algoritmos modernos en aprendizaje automático y optimización.

Tomemos el problema estándar de SVM con la inclusión de una transformación $\phi(\cdot)$ que permita tipos de fronteras más allá de las lineales (3.89). Llamamos

$$J_1 = \min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_i(w^T \cdot \phi(x_i) + b) \geq 1 \quad \forall\, 1 \leq i \leq n_{\text{tr}} \tag{3.91}$$

Dicho problema puede reescribirse usando *multiplicadores de Lagrange* α_i (para mayor información sobre el tema ver [12, Sección 13.7] y [19, Capítulo 5]):

$$J_1 = \min_{w,b} \max_{\alpha_i \geq 0} \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n_{\text{tr}}} \alpha_i [y_i(w^T \cdot \phi(x_i) + b) - 1] \tag{3.92}$$

La equivalencia entre (3.91) y (3.92) no es trivial, y requiere una explicación adicional. Es importante notar que si se cumple la condición $y_i(w^T \cdot \phi(x_i) + b) > 1$ los multiplicadores que maximicen la expresión anterior deberán ser nulos $\alpha_i = 0$, y si en cambio, la condición se da con igualdad $y_i(w^T \cdot \phi(x_i) + b) = 1$ el multiplicador α_i sería irrelevante ya que estaría multiplicado por cero. Llegado el caso de que todas las muestras tengan alguna de estas dos característica, las expresiones matemáticas (3.91) y (3.92) coinciden. Esto ocurrirá absolutamente siempre, y dicha conclusión se puede razonar por el absurdo. Supongamos un par (w, b) que lograra que al menos una de las muestras cumpla $y_i(w^T \cdot \phi(x_i) + b) < 1$; esto obligaría a ese multiplicador $\alpha_i \to \infty$ y por lo tanto $J_1 \to \infty$. Esto nunca se elegirá pues se busca el (w, b) que minimicen la expresión. Por lo tanto, ambas expresiones son equivalentes.

Una importante característica a resaltar es que el multiplicador α_i funciona como detector de vectores soportes. Si las muestras cumplen $\alpha_i > 0$ necesariamente deben ser vectores soportes [10].

La solución del problema primal J_1 define un problema de optimización del tipo mín − máx

[10] Por cuestiones numéricas, suele traer complicaciones detectar vectores soportes como $\alpha_i > 0$. En la práctica suele compararse $\alpha_i > \epsilon$ con $\epsilon > 0$ un número pequeño.

con Lagrangiano[11]:

$$\mathcal{L}\left([w,b],[\alpha_1,\cdots,\alpha_{n_{\text{tr}}}]\right) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n_{\text{tr}}} \alpha_i[y_i(w^T\cdot\phi(x_i)+b)-1] \tag{3.93}$$

En este contexto, definimos su problema dual como un problema del tipo máx − mín para el mismo Lagrangiano:

$$J_2 = \max_{\alpha_i\geq 0}\min_{w,b}\ \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n_{\text{tr}}} \alpha_i[y_i(w^T\cdot\phi(x_i)+b)-1] \tag{3.94}$$

La relación entre J_1 y J_2 es un problema habitual en la teoría de optimización. En particular, el siguiente teorema vincula ambas problemáticas.

Propiedades 3.5 (Weak and Strongh duality) *Sean J_1 y J_2 definidos como*

$$J_1 = \min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}}\ f(x,y), \qquad J_2 = \max_{y\in\mathcal{Y}}\min_{x\in\mathcal{X}}\ f(x,y) \tag{3.95}$$

Luego $J_1 \geq J_2$. En particular, para el problema de SVM con J_1 (3.92) y J_2 (3.94), si las clases son linealmente separables de forma estricta (con un posible margen no nulo) vale con igualdad $J_1 = J_2$.

Demostración 3.6 (Prop. 3.5) *Para la primera afirmación, notar que para todo par $(x',y')\in\mathcal{X}\times\mathcal{Y}$:*

$$\min_{x\in\mathcal{X}}\ f(x,y') \leq f(x',y') \leq \max_{y\in\mathcal{Y}}\ f(x',y) \tag{3.96}$$

Como la desigualdad vale para cualquier par (x',y'), basta con tomar los extremos de dicha inecuación y elegir el x' que minimice y el y' que maximice:

$$J_2 = \max_{y'\in\mathcal{Y}}\min_{x\in\mathcal{X}}\ f(x,y') \leq \min_{x'\in\mathcal{X}}\max_{y\in\mathcal{Y}}\ f(x',y) = J_1 \tag{3.97}$$

La segunda parte de la demostración se conoce como dualidad fuerte *y su demostración escapa a las pretensiones del presente manuscrito. Se recomienda ver [19, Capítulo 5].*

Al intercambiar al mínimo con el máximo, el problema se simplifica notoriamente centrándose, en primer lugar, en la optimización con respecto a los parámetros (w,b). En primer lugar se plantea considerar fijos los multiplicadores α_i y minimizar el Lagrangiano (3.93), igualando a cero sus derivadas:

$$\frac{\partial}{\partial w}\mathcal{L}\left([w,b],[\alpha_1,\cdots,\alpha_{n_{\text{tr}}}]\right) = w - \sum_{i=1}^{n_{\text{tr}}} \alpha_i y_i \phi(x_i) \tag{3.98}$$

$$\frac{\partial}{\partial b}\mathcal{L}\left([w,b],[\alpha_1,\cdots,\alpha_{n_{\text{tr}}}]\right) = -\sum_{i=1}^{n_{\text{tr}}} \alpha_i y_i \tag{3.99}$$

[11]Se denomina Lagrangiano a la función resultante a optimizar una vez incorporados las restricciones del problema via multiplicadores de Lagrange.

Sean w^* y b^* los valores óptimos, resultado de igualar a cero las derivadas anteriores. Mientras que de la primera ecuación se deduce el valor de $w^* = \sum_{i=1}^{n_{\text{tr}}} \alpha_i y_i \phi(x_i)$, la segunda impone una nueva condición $\boldsymbol{\alpha}^T \mathbf{y} = 0$ con $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n)$ y $\mathbf{y} = (y_1, \cdots, y_n)$. Utilizando estas dos identidades, el Lagrangiano se puede simplificar como:

$$\mathcal{L}\left([w^*, b^*], [\alpha_1, \cdots, \alpha_n]\right) = \frac{1}{2}\|w^*\|^2 - w^{*T} \cdot \sum_{i=1}^{n} \alpha_i y_i \phi(x_i) - b \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i \tag{3.100}$$

$$= \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2}\|w^*\|^2 \tag{3.101}$$

$$= \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) \tag{3.102}$$

$$= \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha} \tag{3.103}$$

donde $\mathbf{Q} \in \mathbb{R}^{n_{\text{tr}} \times n_{\text{tr}}}$ es una matriz cuyos coeficientes son $Q_{i,j} = y_i y_j \phi(x_i)^T \phi(x_j)$. Finalmente, J_2 puede escribirse como:

$$J_2 = \max_{\boldsymbol{\alpha}} \ \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha} \quad \text{s.t.} \quad \boldsymbol{\alpha}^T \mathbf{y}, \quad \alpha_i \geq 0 \quad \forall\, 1, \cdots, n_{\text{tr}} \tag{3.104}$$

Una característica importante a resaltar es el hecho de que el problema pueda reducir su complejidad mediante el precómputo de los vectores soporte, ya que el único término a computar previo a la optimización es $\mathbf{Q}$. Sea $\mathcal{S}$ el conjunto de índices de vectores soportes, $\alpha_i = 0$ para todo $i \notin \mathcal{S}$ y por lo tanto:

$$\boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha} = \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{3.105}$$

donde $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ se denomina **kernel** $k : \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \to \mathbb{R}$. Esto implica que para entrenar solo es necesario computar esta magnitud entre los vectores soportes. Además, el planteo dual permite generalizar el problema a cualquier tipo de kernel: no es relevante conocer ni el valor de $\phi(x)$ ni el valor de w^*, pudiendo estos tener dimensión infinita y nunca ser computados. Esto ocurre también para el cómputo del bias b^*, ya que para todo $i \in \mathcal{S}$ debe ocurrir que $y_i(w^{*T}\phi(x_i) + b^*) = 1$. En teoría bastaría conocer un solo vector soporte para poder despejar b^*, pero en la práctica se suele tomar el promedio para mitigar errores numéricos:[12]

$$b^* = \frac{1}{N_S} \sum_{i \in \mathcal{S}} \left(y_i - w^{*T} \phi(x_i)\right) = \frac{1}{N_S} \sum_{i \in \mathcal{S}} \left(y_i - \sum_{j \in \mathcal{S}} \alpha_j y_j k(x_i, x_j)\right) \tag{3.106}$$

donde N_S representa la cantidad de vectores soportes. Este comportamiento también se da durante las predicciones (validación, testeo, etc.), ya que la clasificación tanto a nivel

[12]Notar que $\frac{1}{y_i} = y_i$ ya que $y_i \in \{-1, 1\}$.

soft $z(x)$ como a nivel *hard* $\varphi(x) = \text{SIGNO}[z(x)]$ cumple:

$$z(x) = w^{*T}\phi(x) + b^* = \sum_{i=1}^{n_{\text{tr}}} \alpha_i y_i \phi(x_i)^T \phi(x) + b^* = \sum_{i \in \mathcal{S}} \alpha_i y_i k(x_i, x) + b^* \tag{3.107}$$

donde solamente es necesario computar el kernel entre el vector a predecir y los vectores soportes. Este tipo de estructuras donde solamente se necesitan computar el kernel en pocas muestras se denomina *sparse* [20, Capítulo 7]. Si bien el problema primal puede o no ser más difícil (computacionalmente) de resolver que el dual, este último tiene la ventaja de tener su estructura sparse (y por lo tanto aliviar las predicciones) y de poder utilizar funciones $\phi(\cdot)$ de dimensión infinita (ya que no necesitan ser calculadas).

Algunos de los kernel más utilizados en la práctica son:

- Lineal: $k(x, x') = x^T x'$.
- Polinómico: $k(x, x') = \left(\gamma \cdot x^T x' + r\right)^d$ (con valores prefijados de γ, r y d).
- RBF o gaussiano: $k(x, x') = e^{-\gamma \|x - x'\|^2}$ (con γ prefijado).

3.7. Árboles de Decisión

Es habitual en la bibliografía referirse a la mayoría de los algoritmos de inteligencia artificial como *cajas negras*. Esta idea está relacionada con preguntarse sobre como un algoritmo *razona*, y por lo tanto va de la mano con el tipo de terminología usada en el área. Desde términos clásicos en la temática como aprendizaje, entrenamiento o neurona hasta más modernos como inteligencia artificial o mecanismo de atención, el *marketing* de los algoritmos trata de humanizar los diferentes procedimientos presentes en los mismos. Reflexionar sobre esta característica no solo deja de manifiesto que la inteligencia artificial no tiene tanto de inteligencia como uno puede suponer, sino que quizás tampoco tiene mucho de artificial. No es para nada artificial la inversión en tecnología que recibió el área en las últimas décadas, ya que el exceso de humanización en los términos usados refiere a una búsqueda por cierta parte de la sociedad. Además de ser una nueva pieza de tecnología es un negocio muy rentable para algunas personas.

Intentando dejar de lado la parte semántica de la cuestión, tiene algo de cierto que a pesar de conocer perfectamente el riesgo empírico minimizado durante el entrenamiento por los algoritmos, no contamos con una interpretación conceptual sobre la relación entre los valores de los parámetros y las reglas de decisión resultante. En este sentido utilizaremos la idea de caja negra, siempre teniendo en cuenta lo relativo de los términos que utilizamos para referirnos a los diferentes procesos. Utilizar un algoritmo para resolver una tarea es importante, pero imaginemos el potencial de abrir el algoritmo y entender

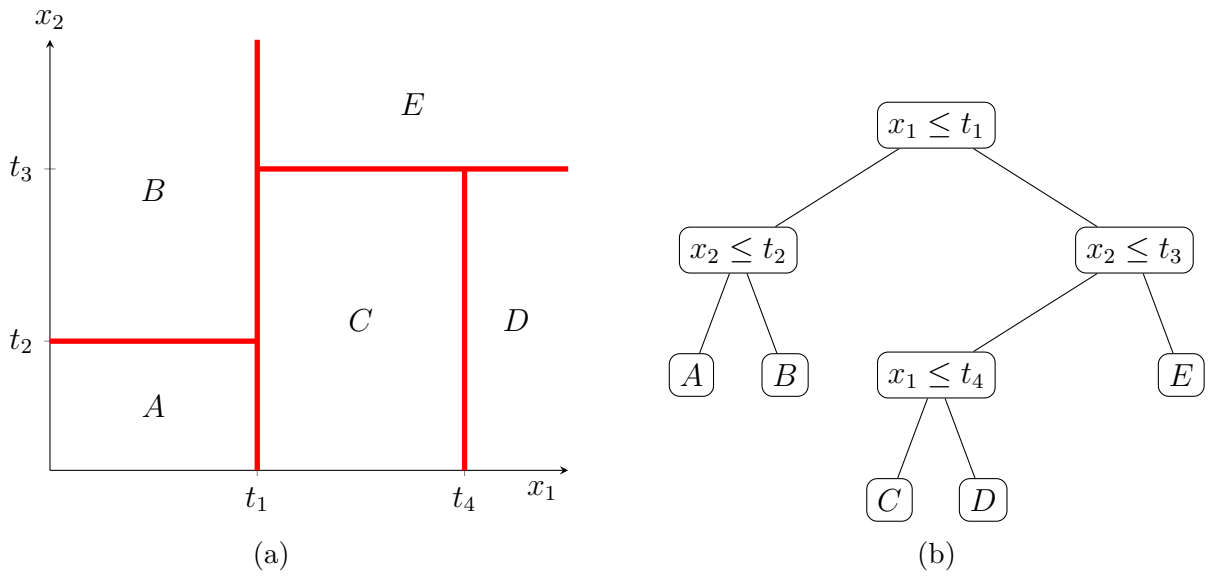


Figura 3.12: Gráficos CART. (a) Frontera de decisión; y (b) diagrama de árbol (por convención, si la condición se cumple se va hacia la izquierda).

los por menores de su razonamiento. Si hay algo de emular el comportamiento humano en la inteligencia artificial, quizás mirando el algoritmo nos veamos a nosotros mismos.

Con esta idea es que surgen los *arboles de clasificación y regresión* (CART, por sus siglas en inglés Clasification And Regression Trees). Restringiendo las posibles reglas de decisión a cuadrículas, como puede verse la Fig. 3.12a, el algoritmo resultante puede graficarse con un diagrama de árbol donde cada nodo puede expresarse como la comparación de uno de los predictores (x_1 o x_2 en este caso) contra un umbral. Un posible diagrama de árbol para la frontera de decisión mencionada[13] puede verse en la Fig. 3.12b, donde por convención se interpreta que satisfacer la condición implica un desplazamiento hacia la izquierda.

Se denomina raíz al nodo originario del árbol (en la Fig. 3.12b es el nodo que evalúa si $x_1 \leq t_1$), y hojas a los nodos terminales (A, B, C, D y E en este ejemplo). El entrenamiento consiste en aprender, para cada nodo m que no sea hoja, el índice j_m del predictor y el valor del umbral t_m para el cuál se comparará $x_{j_m} \leq t_m$. Una vez entrenado, la decisión final en un problema de clasificación consistirá en elegir la clase más frecuente en entrenamiento para cada hoja. En el caso de una tarea de regresión, el valor promedio de las etiquetas de cada hoja.

Matemáticamente, el problema de optimización a resolver durante el entrenamiento puede modelarse calculando j_m y t_m para cada nodo. Sea Q_m al conjunto de datos supervisado de entrenamiento presentes en el nodo $m-$ésimo, se definen los subconjuntos de

[13]Cada árbol representa una frontera, pero cada regla de decisión puede graficarse con varios diagramas de árbol. La relación no es biyectiva.

datos a izquierda $Q_m^L(j_m, t_m)$ y $Q_m^R(j_m, t_m)$ como:

$$Q_m^L(j_m, t_m) = \{(x, y) \in Q_m : \ x_{j_m} \leq t_m\} \tag{3.108}$$

$$Q_m^R(j_m, t_m) = \{(x, y) \in Q_m : \ x_{j_m} > t_m\} \tag{3.109}$$

Con estos subconjuntos se define el problema de optimización a resolver $(j_m^*, t_m^*) = \arg\min_{j_m, t_m} G_m(j_m, t_m)$, con

$$G_m(j_m, t_m) = \frac{\#\left(Q_m^L(j_m, t_m)\right)}{\#\left(Q_m\right)} H(Q_m^L(j_m, t_m)) + \frac{\#\left(Q_m^R(j_m, t_m)\right)}{\#\left(Q_m\right)} H(Q_m^R(j_m, t_m)) \tag{3.110}$$

donde $H(Q)$ es la función impureza del conjunto de datos Q. Es razonable pensar que la impureza a izquierda pese poco si la proporción de muestras hacia ese lado es baja. Notar que $G_m(j_m, t_m)$ es una métrica probabilista del la impureza posterior a la separación del conjunto. Las funciones impurezas mide que tan desconcentrado es un nodo. Es razonable pensar entonces que si todas las muestras presentes en un nodo tienen la misma etiqueta, la impureza sea nula. De la misma manera, la impureza es máxima cuando las proporciones son uniformes. Las funciones más habituales en problemas de clasificación son la entropía y impureza de Gini, definidas como:

- Gini: $H(Q) = \sum_k p_k(1 - p_k)$.
- Entropía: $H(Q) = \sum_k -p_k \log_2(p_k)$.

donde p_k hace referencia a la proporción empírica en el conjunto de datos de la clase $k-$ésima. A continuación se mostrará el comportamiento de estas métricas para un problema de clasificación binaria.

Ejemplo 3.6 *Para un conjunto de datos de dos clases de proporciones p y $1 - p$, graficar las diferentes impurezas en función de p.*

- **Impureza de Gini:** En este caso, la función impureza toma el valor $H(Q) = p(1 - p) + (1 - p)p = 2p(1 - p)$. Es una parábola cóncava que alcanza su valor máximo en $p = \frac{1}{2}$ y vale $\frac{1}{2}$. El gráfico puede verse en la Fig. 3.13a.
- **Entropía:** En este caso, la función impureza puede escribirse, usando logartimos naturales, como $H(Q) = \frac{-p\log(p)-(1-p)\log(1-p)}{\log(2)}$. Derivando esta función se observa que $\frac{dH(Q)}{dp} = \frac{-\log(p)+\log(1-p)}{\log(2)}$ y $\frac{d^2H(Q)}{dp^2} = -\frac{1}{\log(2)}\left(\frac{1}{p} + \frac{1}{1-p}\right) < 0$. Se observa también un comportamiento cóncavo (segunda derivada negativa) cuyo valor máximo se alcanza (cuando la derivada se anula) en $p = \frac{1}{2}$ y vale 1. El gráfico puede verse en la Fig. 3.13b.

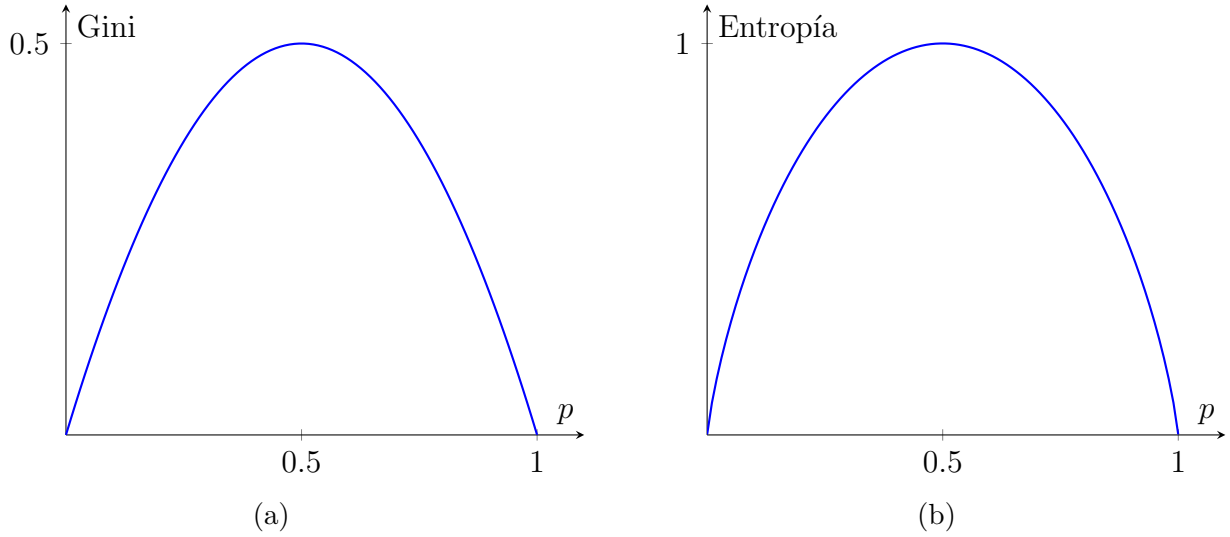


Figura 3.13: Funciones impureza para un problema de clasificación binaria. (a) Impureza de Gini; y (b) Entropía.

La implementación habitual de este tipo de algoritmos suele encontrar el óptimo valor de $G_m(j_m, t_m)$ por prueba y error. Cabe resaltar que este algoritmo no utiliza gradiente descendente, por lo que no posee problemas de convergencia. Además, al comparar de a un predictor por vez, no tiene sentido normalizar los datos para este tipo de algoritmos.

En el caso de un problema de regresión, el procedimiento es similar. Un árbol para este tipo de problemas asume una función regresión constante por regiones. Las predicciones se efectúan asignando el valor promedio de la hoja y la impureza utilizada es el error cuadrático medio:

$$H(Q_m) = \sum_{(x,y)\in Q_m} (y - \varphi(x))^2 \tag{3.111}$$

con $\varphi(x)$ el valor promedio de las $y-$s correspondientes a la hoja a la que pertenece x.

Una de las características más interesantes de los árboles, es que nos permite ponderar a los predictores pudiendo identificar cuales son importantes y cuales no. Esta dinámica rompe el fenómeno de *caja negra*, dándonos información sobre que variables son relevantes y cuales pueden ser omitidas. Se denomina *Feature Importance* de un predictor, a la suma de las ganancias (en impureza), de cada nodo (Δ_m) donde se haya utilizado dicho predictor para separar el conjunto con $\Delta_m = H(Q_m) - G_m(j_m^*, t_m^*)$. Habitualmente el resultado se expresa normalizado para que sumen 1.

Otra característica importante a tener en cuenta es cuando finalizar el árbol. Se llama **árbol maximal** al desarrollo del árbol hasta que todas las etiquetas del conjunto de entrenamiento de cada hoja sean iguales. Básicamente cuando un nodo posee impureza nula se lo interpreta como hoja. El problema con los árboles maximales es el *overfitting*, ya que genera reglas muy dependientes del conjunto de entrenamiento. Alternativas para

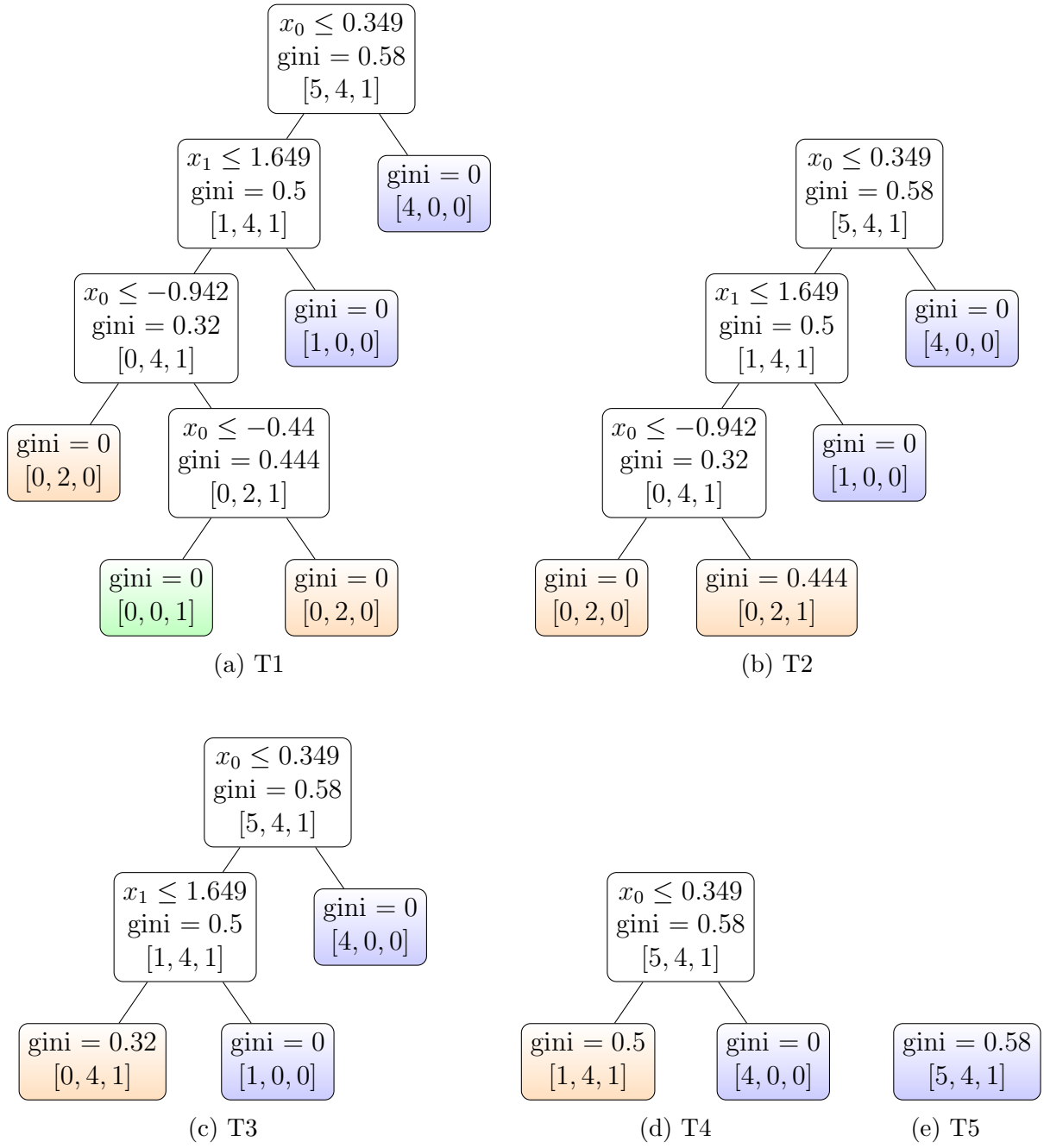


Figura 3.14: Ejemplo de podado de árbol. (a) T1: Árbol maximal de 5 hojas, (b) T2: subárbol de 4 hojas, (c) T3: subárbol de 3 hojas, (d) T4: subárbol de 2 hojas, (e) T5: subárbol de 1 hoja.

evitar este problema es preestablecer la profundidad máxima o prohibir la separación en nodos con pocas muestras (menor a un número prefijado). Sin embargo, la mejor manera de mitigar este problema es *podando* el árbol maximal.

3.7.1. Podado

Sea T un árbol determinado, $L(T)$ su respectivo conjunto de hojas y α el hiperparámetro de complejidad (multiplicador que regulará la poda). Se denomina medida de costo-complejidad a

$$H_\alpha(T) = \sum_{m \in L(T)} \frac{\#(Q_m)}{n_{\text{tr}}} \cdot H(Q_m) + \alpha \cdot \#(L(T)) \tag{3.112}$$

La poda se basa en quedarse con el subárbol del árbol maximal de menor costo-complejidad. Básicamente, (3.112) consiste en definir un costo regularizado, tomando como término principal la impureza de las hojas (suma ponderada teniendo en cuenta la proporción) e incorporando una penalización a los árboles grandes. Para fijar ideas analicemos un ejemplo concreto.

Ejemplo 3.7 *Sea el árbol de decisión maximal presentado en la Fig. 3.14a.*

1. *Indicar todos los posibles subárboles.*
2. *Indicar el costo-complejidad para cada subárbol, como función de* α.
3. *Elegir el subárbol de menor costo-complejidad para* $\alpha = 0.1$ *e indicar la* feature importance.

En la Fig. 3.14a se muestra un árbol maximal para una tarea de clasificación de 3 clases. En colores se resaltan las hojas, utilizando un color diferente para la clase mayoritaria según corresponda. Se indica condición de separación, impureza de Gini y cantidad de muestras que llegan al nodo ($n_{\text{tr}} = 10$). Mientras que el árbol maximal T1 posee 5 hojas, en las Figs. T2 3.14b, T3 3.14c, T4 3.14d y T5 3.14e representan subárboles de 4, 3, 2 y 1 hoja respectivamente. En total hay 5 opciones de árboles (se analizaron todos los subárboles posibles), y se buscará elegir el de menor costo-complejidad. Teniendo en cuenta la proporción de muestras de cada hoja, el costo complejidad $H_\alpha(T)$ toma los siguientes valores dependiendo del valor de α :

- $H_\alpha(T_1) = 0 + 5\alpha$.
- $H_\alpha(T_2) = 0.13 + 4\alpha$.
- $H_\alpha(T_3) = 0.16 + 3\alpha$.

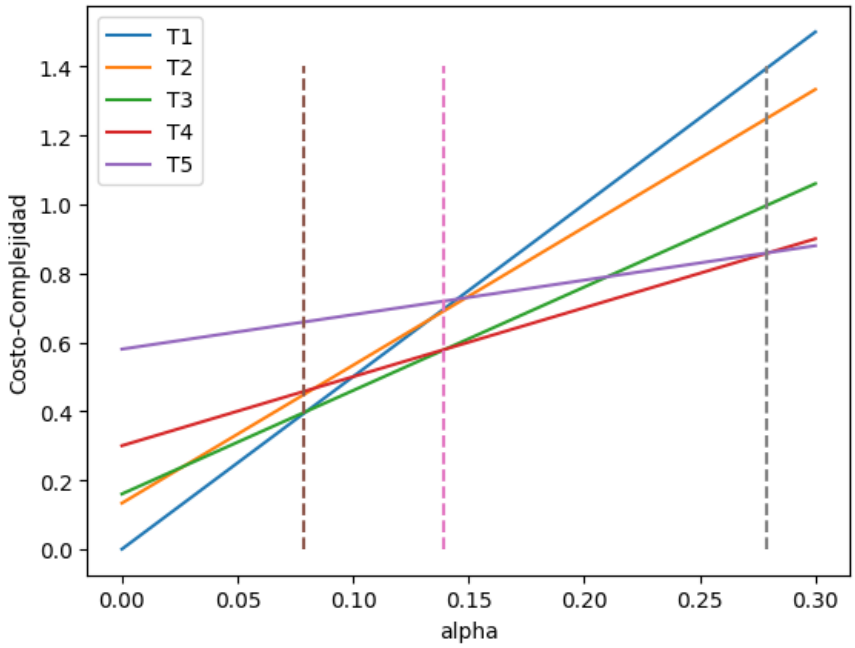


Figura 3.15: Costo-complejidad en función del hiperparámetro de complejidad α para los diferentes subárboles.

- $H_\alpha(T_4) = 0.3 + 2\alpha$.
- $H_\alpha(T_5) = 0.58 + \alpha$.

Mientras que el árbol maximal T1 no posee impurezas, recibe la mayor penalización debido a su tamaño. Este será el indicado para valores de α muy bajos. En contraste, el subárbol T5 de un solo nodo posee la penalización más baja, pero su impureza se corresponde a la del nodo raíz. Será el indicado para valores de α muy grandes. En la Fig. 3.15 puede verse las diferentes funciones costo-complejidad $H_\alpha(T)$ en función del hiperparámetro de complejidad α para los diferentes subárboles.

Es interesante notar como el subárbol T2 no minimiza el costo-complejidad para ningún valor de α. Esto mismo ocurre siempre que hay dos subárboles distintos con la misma cantidad de hojas (las rectas son paralelas). Eso quiere decir que, detectándolo de antemano, se puede reducir la cantidad de candidatos a subárbol óptimo. Esto reduce significativamente el costo computacional de una potencial etapa de validación.

En particular, para $\alpha = 0.1$ el árbol óptimo es T3. Para el cómputo de la *feature importance* notar que hay un solo nodo de separación utilizando x_0 (se llamará nodo 0) y uno solo utilizando x_1 (se llamará nodo 1). Sus impurezas son 0.58 y 0.5 respectivamente. Finalmente se calcula las ganancias:

$$\Delta_0 = H(Q_0) - G_0(0, 0.349) = 0.58 - \left(\frac{6}{10} \cdot 0.5 + \frac{4}{10} \cdot 0\right) = 0.28 \tag{3.113}$$

$$\Delta_1 = H(Q_1) - G_1(1, 1.649) = 0.5 - \left(\frac{5}{6} \cdot 0.32 + \frac{1}{6} \cdot 0\right) = 0.233 \tag{3.114}$$

Normalizando para que sumen 1, la importancia de cada predictor es 0.546 y 0.454 respectivamente.

3.7.2. Bosques Aleatorios

Los árboles de decisión no son solamente algotimos cuya función es romper el esquema de caja negra, sino que fueron *estado de arte* en muchas aplicaciones durante mucho tiempo, incluso en algunas tareas lo siguen siendo. Su limitante principal son los problemas de *overfitting*, que incluso con podando puede ser muy relevante. Es entonces cuando surge la idea de sacrificar interpretabilidad de resultados en pos de mejorar el desempeño.

Los bosques aleatorios son un algoritmo capaz de logar esto. Están basados en un principio llamado *Bagging*: Entrenar múltiples algoritmos y decidir por mayoría o promedio (en clasificación o regresión respectivamente). Un algoritmo de múltiples árboles se llama bosque. Sean $Y_1, \cdots, Y_B$ la predicción de $B-$algoritmos de la misma media $\mathbb{E}[Y_b] = \mu$ y la misma varianza $\text{var}(Y_b) = \sigma^2$, se puede notar que:

$$\mathbb{E}\left[\frac{1}{B}\sum_{b=1}^{B} Y_b\right] = \mu, \qquad \text{var}\left(\frac{1}{B}\sum_{b=1}^{B} Y_b\right) = \frac{\sigma^2}{B} \tag{3.115}$$

El promediado de resultados permite mantener el resultado medio mientras se reduce la varianza[14].

Una segunda cuestión a tener en cuenta a la hora de utilizar bosques, es como aleatorizar los algoritmo de manera que no se esté promediando B árboles iguales. Habitualmente los árboles aleatorios lo hace con dos decisiones aleatorias. La primera consiste en no utilizar todos los d_x predictores en todos los árboles sino que seleccionar al azar aproximadamente $\sqrt{d_x}$ por árbol, fomentando así que los árboles resultantes sean distintos. La segunda aleatoridad consiste en utilizar una técnica llamada **Boostrap**.

Boostrap consiste en utilizar datos diferentes en cada árbol. Para ello se generan B conjuntos de datos del mismo tamaño que el conjunto de datos original n_{tr}. La manera de armar estos conjuntos es eligiendo al azar n_{tr} datos del conjunto original *con reposición*. Es interesante notar que la probabilidad de que un dato particular no forme parte de un conjunto es de $\approx 37\,\%$. Esto puede observarse aplicando el siguiente límite:

$$\left(1 - \frac{1}{n_{\text{tr}}}\right)^{n_{\text{tr}}} \to e^{-1} \tag{3.116}$$

[14]En clasificación, si se piensan etiquetas en codificación *one-hot*, promediar para luego elegir el máximo equivale a elegir la respuesta mayoritaria.

4

Aprendizaje no Supervisado

Los datos son la materia prima de la inteligencia artificial. Sacarle el mayor provecho posible a los mismos es el objetivo. No se puede dar el lujo de desperdiciar nada. Ahí radica el verdadero potencial de estos algoritmos.

En los Capítulos 2 y 3 se han estudiado algoritmos de **aprendizaje supervisado**. Es decir, dado un conjunto de datos $\{(x_i, y_i)\}_{i=1}^{n_{\text{tr}}}$ entrenar un modelo que permita inferir Y a partir de X. El problema con este enfoque es que no permite utilizar los datos x que no tengan asociados su correspondiente etiqueta y. En la era de la información, el procesamiento de la misma puede ser la clave para la maduración económica competitiva de diferentes aplicaciones e industrias en la ciencia de datos. Cada pieza de información posee un valor intrínseco que no debe ser desaprovechado, lo que vuelve a las bases de datos cada vez más grandes. El problema es que la mayor parte de los datos disponibles no están etiquetados, y dependiendo de la aplicación, el etiquetado puede ser muy costoso. En ese sentido es esencial utilizar los datos sin etiquetar.

Llamamos **aprendizaje no supervisado** al aprendizaje efectuado a partir de un conjunto de datos no etiquetados $\{x_i\}_{i=1}^{n_{\text{tr}}}$. Por si solo es muy pretencioso pretender información sobre una variable Y no observable. Sin embargo, eso no quita que no pueda ser beneficioso inferir algunas características de X. En particular, se enmarcará el *aprendizaje no supervisado* en la tarea denominada **autoencoder** [21, Capítulo 14].

El *autoencoder* consiste en dos algoritmos, entrenados habitualmente en simultáneo, llamados **encoder** o *codificador* y **decoder** o *decodificador*. Básicamente las entradas $X \in \mathbb{R}^{d_x}$ se codifican a un **espacio latente** $Z \in \mathcal{Z}$ para luego ser decodificados y estimar la variable original $\hat{X} \in \mathbb{R}^{d_x}$. En el caso de que el espacio latente $\mathcal{Z}$ sea finito, el problema recibe el nombre de **clustering** o *agrupamiento*. En cambio si $\mathcal{Z} = \mathbb{R}^{d_z}$ con $d_z < d_x$ hablamos de un problema de **manifold learning** o reducción de dimensión.

Cabe destacar que el objetivo del autoencoder no es simplemente reconstruir una variable (que de por si es observable), sino obtener información sobre la distribución de los datos, sus características y secretos. En ese sentido, el aprendizaje no supervisado busca entender los datos. En líneas generales, los autoencoders tienen tres tipos de aplicaciones:

- Preprocesar los datos para una posterior tarea de regresión o clasificación. Por ejem-

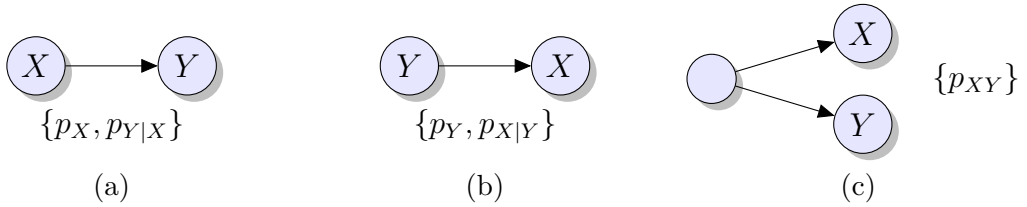


Figura 4.1: Relaciones causales entre variables. (a) Una relación causal; (b) una anticausal; y (c) una relación no causal.

plo, utilizar como entrada de la segunda tarea la variable latente o la reconstrucción de un autoencoder.

- Resolver un problema específico *no supervisado* por si mismo. Por ejemplo compresión de datos con pérdida de información o un detector de anomalías.
- Interpretar un conjunto de datos. Por ejemplo reducir el espacio de predictores a dos dimensiones para poder visualizarlos o reducirlos a una dimensión para representarlos con un solo número.

A continuación se analizará la primera de estas aplicaciones. En este problema se necesitan tanto datos no supervisados para entrenar el autoencoder como datos supervisados para entrenar el regresor/clasificador. Este paradigma se conoce como **aprendizaje semisupervisado** y es de lo más habitual en la práctica: se cuenta con muchos datos sin etiquetar y unos pocos etiquetados. Pero es importante destacar que mientras que el objetivo del algoritmo supervisado es inferir (parcial o totalmente) la distribución de $p_{Y|X=x}(y)$, la del algoritmo no supervisado es hacer lo propio con $p_X(x)$. Mejorar el conocimiento de la distribución marginal de X puede o no ser relevante para inferir la condicional $Y|_{X=x}$. A continuación se analizará que condiciones deben cumplirse para considerer relevante una etapa no supervisada para una posterior regresión/clasificación.

4.1. Causalidad en Aprenizaje Semisupervisado

Determinar si el conocimiento de $p_X(x)$ ayuda a inferir $p_{Y|X=x}(y)$ escapa al problema probabilístico. En la teoría de probabilidad, cualquier conjunta se puede construir multiplicando $p_{XY}(x, y) = p_{Y|X=x}(y)p_X(x)$, denotando que pueden tener naturalezas totalmente diferentes. Para saber si este es el caso o no, hay que adentrarse más profundamente en el problema y analizar el modelo físico subyacente que generó las variables. Es necesario adentrarse en la **teoría de causalidad**.

La cantidad de lluvia y la cantidad de paraguas abiertos en una plaza son variables estadísticamente correlacionadas. Pero mientras que aumentar el nivel de la lluvia suele

aumentar la cantidad de paraguas, aumentar la cantidad de paraguas abiertos no aumenta el nivel de lluvia. En este ejemplo la cantidad de lluvia es la causa y la cantidad de paraguas es el efecto, definiendo una relación lluvia $\rightarrow$ paraguas. Estudiar la relación de causalidad entre las variables, nos permite caracterizar los mecanismos que las generaron. Pero en general requiere interpretar a las mismas a partir de su física, más allá de la estadística.

El **principio de independencia de mecanismos causales**[1] dice que, el proceso causal generativo de las variables de un sistema se compone de mecanismos autónomos que no se informan ni influyen entre sí. En el caso probabilístico, esto significa que la distribución condicional de cada variable, dadas sus causas (es decir, su mecanismo), no informa ni influye en los demás mecanismos [22, Capítulo 2]. Este principio pone el foco en entender el proceso de generación de las variables, tal como se programaría la fuente en el caso de necesitar generar datos sintéticos.

La Fig. 4.1 menciona todas las relaciones de causalidad posibles entre los datos X y las etiquetas Y. Se denomina relación causal, Fig. 4.1a a la estructura donde $X \rightarrow Y$. En esta se definen como mecanismos independientes $p_X(x)$ y $p_{Y|X=x}(y)$, descartando la posibilidad de que una etapa no supervisada ayude a resolver un problema supervisado. Se denomina relación anticausal, Fig. 4.1b a la estructura donde $Y \rightarrow X$. En esta se definen como mecanismos independientes $p_Y(y)$ y $p_{X|Y=y}(x)$, denotando una posible influencia entre $p_X(x)$ y $p_{Y|X=x}(y)$. Lo mismo ocurre en las relaciones no causales, Fig. 4.1c, donde no hay una causalidad entre las variables.

El problema de este tipo de análisis radica en la dificultad de catalogar la relación de causalidad. Es entonces, cuando surge la idea de intentar deducir estas relaciones a partir de la estadística. Básicamente queremos corroborar si el modelo se corresponde a alguno de los siguientes:

- Modelo causal: Se genera $X \sim p_X$, y luego $Y = g(X, U)$ con $U \sim p_U$ independiente a X. Este modelo se caracteriza con p_U y $g(x, u)$.
- Modelo anticausal: Se genera $Y \sim p_Y$, y luego $X = h(Y, V)$ con $V \sim p_V$ independiente a Y. Este modelo se caracteriza con p_V y $h(y, v)$.

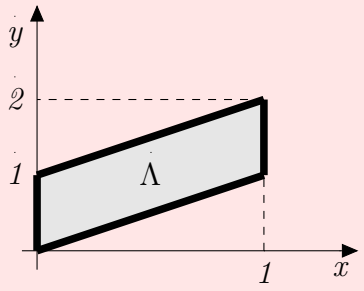
La identificación de estos modelos es imposible, ya que para cualquier conjunta $p_{XY}(x, y)$ siempre existe tanto un modelo causal p_U y $g(x, u)$ como un modelo anticausal p_V y $h(y, v)$. Esto se demuestra con la Prop. 1.9: Sea $U \sim \mathcal{U}(0, 1)$ y $g(x, u) = F^{-1}_{Y|X=x}(u)$, la variable $Y|_{X=x} = g(x, U) \sim p_{Y|X=x}(y)$. Nuevamente caemos en la cuenta de que la causalidad está determinada por el modelo físico, va más allá de la estadística.

Sin embargo, los probabilistas somos obstinados. Si bien nunca se podrá asegurar la relación de causalidad observando la estadística, si se puede presuponer. Es decir, si bien

[1] No confundir independencia estadística con independencia de mecanismos. La primera estudia la influencia entre variables aleatorias y la segunda la influencia entre las distribuciones.

no se tiene certeza, puede existir evidencia hacia un lado o hacia otro. La suposición en este tipo de análisis es que los modelos mas simples son más habituales que los complejos. Por lo tanto, si uno de los modelos es considerablemente más sencillo que el otro, se puede suponer que es el correcto. Este es el caso típico de la mezcla, donde simular primero la variable mezcladora categórica es mucho más sencillo que hacerlo al final. Otros ejemplos serán analizados a continuación.

Ejemplo 4.1 *Estudiar la relación de causalidad para* $(X, Y) \sim \mathcal{U}(\Lambda)$*, donde* Λ *es de la forma:*



En este tipo de ejercicios tenemos que analizar si es más sencillo descomponer la distribución conjunta como $p_{XY}(x,y) = p_{Y|X=x}(y)p_X(x)$ o como $p_{XY}(x,y) = p_{Y|X=x}(y)p_X(x)$. Este ejemplo en particular caracteriza las distribuciones condicionales como:

$$Y|_{X=x} \sim \mathcal{U}(x, x+1) \equiv x + \mathcal{U}(0,1), \qquad X|_{Y=y} \sim \begin{cases} \mathcal{U}(0,y) & 0 < y < 1 \\ \mathcal{U}(y-1,1) & 1 < y < 2 \end{cases} \tag{4.1}$$

En este caso, $Y|_{X=x}$ posee una estructura muy sencilla, mientras que $X|_{Y=y}$ requiere hacer una definición partida. Podemos suponer que corresponde a una relación causal $X \to Y$, y el modelo está caracterizado por $Y = g(X,U) = X + U$ con $U \sim \mathcal{U}(0,1)$.

Ejemplo 4.2 *Estudiar la relación de causalidad para*

$$p_{XY}(x,y) = e^{-x}\mathbf{1}\{0 < y < x\} \tag{4.2}$$

En el Ej. 1.1 se dedujo que $Y|_{X=x} \sim \mathcal{U}(0,x) \equiv x \cdot \mathcal{U}(0,1)$. Por lo tanto, su modelo causal $X \to Y$ es $Y = g(X,U) = X \cdot U$ con $U \sim \mathcal{U}(0,1)$. Sin embargo, también puede efectuarse la descomposición inversa:

$$p_{XY}(x,y) = \underbrace{e^{-(x-y)}\mathbf{1}\{x > y\}}_{p_{X|Y}(x|y)} \underbrace{e^{-y}\mathbf{1}\{y > 0\}}_{p_Y(y)} \tag{4.3}$$

es decir que $X|_{Y=y}$ se distribuye como una exponencial corrida en y, y su modelo anticausal $Y \to X$ es $X = h(Y,V) = Y + V$ con $V \sim \exp(1)$. Ambos modelos son sumamente sencillos y no hay evidencia suficiente para suponer cuál es el modelo real.

4.1.1. Influencia del aprendizaje semisupervisado en modelos causales

En capítulos anteriores se demostró que el resultado óptimo para minimizar el error cuadrático medio es el regresor asociado a la esperanza condicional $\mathbb{E}[Y|X=x]$ (Prop 1.8), el resultado óptimo para minimizar la probabilidad de error es el clasificador bayesiano $\arg\max_y P_{Y|X=x}(y)$ (Prop. 3.1) y el óptimo para la entropía cruzada en la verdadera probabilidad $P_{Y|X=x}$ (Prop. 3.2). En todos los casos las soluciones óptimas dependen solo de la distribución condicional $P_{Y|X=x}(y)$, sin ser relevante la distribución marginal $p_X(x)$. Es por ese motivo que en modelos causales $X \to Y$ (donde $p_X(x)$ y $P_{Y|X=x}(y)$ son mecanismos independientes), una etapa no supervisada que permita estimar $p_X(x)$ no afecta a las distribuciones óptimas.

Mientras que en modelos anticausales o no causales la etapa no supervisada puede ser de gran ayuda para el aprendizaje, en modelos causales no parece recomendable. Sin embargo, aunque marginal, la etapa no supervisada puede tener influencia positiva en modelos causales. Dado que en la práctica los algoritmos no alcanzar la solución óptima, una etapa no supervisada puede ayudar a determinar que solución subóptima es mejor [22, Capítulo 5]. A continuación se estudiarán los tres tipos de funciones costo mencionadas anteriormente, para demostrar a que nos referimos.

Demostración 4.1 (Error cuadrático medio) *Sea $\hat{y} = \varphi(x)$, en (1.16) se demostró la relación entre el error cuadrático medio y su correspondiente error bayesiano:*

$$\mathbb{E}[(Y-\varphi(X))^2] = \mathbb{E}[var(Y|X)] + \mathbb{E}[(\mathbb{E}[Y|X]-\varphi(X))^2] \tag{4.4}$$

El error cuadrático $(\mathbb{E}[Y|X=x]-\varphi(x))^2$ alcanza su valor mínimo para $\varphi(x) = \mathbb{E}[Y|X=x]$. Sin embargo, de no alcanzarse, la diferencia entre el error cuadrático medio y el error bayesiano en (4.4) viene dado por su valor medio. Dicho valor medio se calcula a partir de la marginal $p_X(x)$.

Demostración 4.2 (Probabilidad de error) *Sea $\hat{y} = \varphi(x)$, la relación entre el probabilidad de acierto y su correspondiente error bayesiano se puede representar como:*

$$\mathbb{P}(Y=\varphi(X)) = \mathbb{E}\left[\max_{y\in\mathcal{Y}} P_{Y|X=X}(y)\right] + \mathbb{E}\left[\max_{y\in\mathcal{Y}} P_{Y|X=X}(y) - P_{Y|X=X}(\varphi(X))\right] \tag{4.5}$$

La diferencia $\max_{y\in\mathcal{Y}} P_{Y|X=x}(y) - P_{Y|X=x}(\varphi(x))$ alcanza su valor mínimo para $\varphi(x) = \arg\max_y P_{Y|X=x}(y)$. Sin embargo, de no alcanzarse, la diferencia entre la

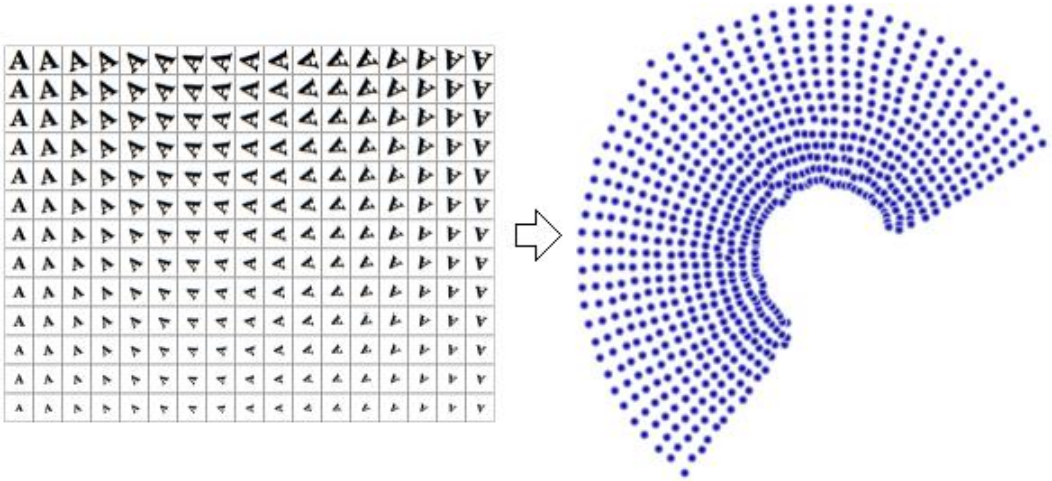


Figura 4.2: Ejemplo de *manifold*, donde los datos pueden caracterizarse con dos valores: módulo y fase. La imagen fue extraída de `https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction`.

probabilidad de acierto y el valor óptimo $\mathbb{E}\left[\max_{y\in\mathcal{Y}} P_{Y|X=X}(y)\right]$ *en* (4.5) *viene dado por su valor medio. Dicho valor medio se calcula a partir de la marginal* $p_X(x)$.

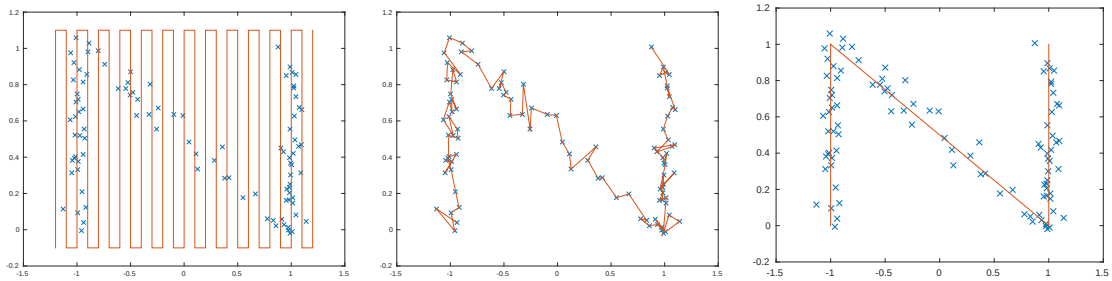
Demostración 4.3 (Entropía Cruzada) *Sea* $\hat{P}(y|x)$*, la predicción* soft *de un clasificador, en* (3.35) *se demostró la relación entre la entropía cruzada y su correspondiente error bayesiano*

$$\mathbb{E}\left[-\log \hat{P}(Y|X)\right] = \mathbb{E}\left[KL\left(P_{Y|X=X}\|\hat{P}(\cdot|X)\right)\right] + \mathbb{E}\left[-\log P_{Y|X=X}(Y)\right] \tag{4.6}$$

La divergencia de Kullback Leibler $KL\left(P_{Y|X=x}\|\hat{P}(\cdot|x)\right)$ *alcanza su mínimo para* $\hat{P}(y|x) = P_{Y|X=x}(y)$*. Sin embargo, de no alcanzarse, la diferencia entre el la entropía cruzada y el error bayesiano en* (4.6) *viene dado por su valor medio. Dicho valor medio se calcula a partir de la marginal* $p_X(x)$.

4.2. Reducción de dimensión y Manifold

Cualitativamente, llamaremos **manifold** o *variedad* al espacio efectivo en el que habitan los datos. Ejemplos de *manifolds* pueden ser curvas en el plano o superficies en el espacio. En la Fig. 4.2 puede verse un ejemplo práctico de este fenómeno. En ésta, se estudia una base de datos con muchos predictores (uno por cada píxel de la imagen) de letras "A". En definitiva, esta base de datos se puede describir mediante dos valores: el tamaño de la letra y el ángulo de inclinación. Es decir que con un $d_z = 2$ es suficiente para explicar la naturaleza de los datos.



(a) (b) (c)

Figura 4.3: Ejemplo de aprendizaje en una tarea de reducción de dimensión [24]. (a) Para un algoritmo con *overfitting*; (b) para uno con *encoder* biyectivo; y (c) para un algoritmo con correcto *manifold learning*.

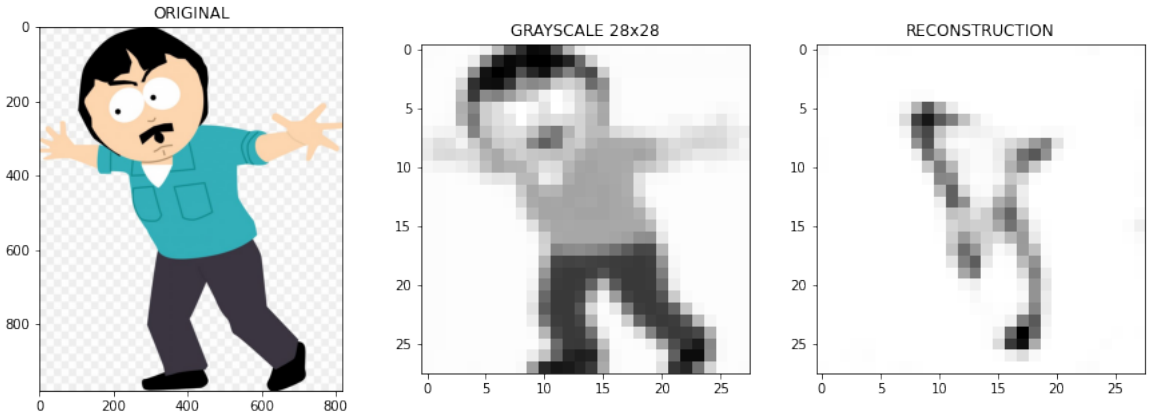
Cabe resaltar que el objetivo de la reducción de dimensión no es simplemente reconstruir los datos. Sino reconstruirlos a partir de una representación relevante para explicar algún fenómeno o resolver otra tarea. Si no se reconocen patrones en la naturaleza de los datos no hay aprendizaje. De hecho, existen transformaciones biyectivas $\mathcal{T} : \mathbb{R}^{d_x} \to \mathbb{R}$ que podrían utilizarse como *encoder* [23, Capitulo 19] (y $\mathcal{T}^{-1}$ como *decoder*). Pero las representaciones latentes obtenidas de esta manera no serán relevantes por no aportar información sobre la distribución de los datos.

La Fig. 4.3a muestra un ejemplo de aprendizaje de un *encoder* capaz de reconstruir cualquier valor. Si el ancho de las líneas es suficientemente fino, todas las muestras caerán sobre la curva, pero la estructura aprendida nada tiene que ver con el *manifold*. La Fig. 4.3b muestra un algoritmo con *overfitting* y la Fig. 4.3c un correcto aprendizaje. En un autoencoder es tan importante no memorizar el conjunto de datos como no aprender una transformación biyectiva, se desea aprender el *manifold*. La regularización de los autoencoders deben balancear estos conceptos.

Cuando el espacio de entrada es de dos dimensiones, como es el caso de la Fig. 4.3, puede visualizarse fácilmente la solución aprendida y corroborar si hay problemas de *overfitting*, *underfitting* o *biyecciones*. Pero en dimensiones más grandes se necesitan métodos más sofisticados. Así como el subajuste se verifica observando el error de entrenamiento y el sobreajuste comparando los errores de entrenamiento y validación, detectar una solución biyectiva requiere otro tipo análisis. Para corroborar si un determinado autoencoder codifica adecuadamente los datos, pero no cualquier tipo de entrada, se utiliza un **detector de anomalías**.

Para ello, se define un segundo conjunto de testeo pero de características anómalas, es decir, un conjunto de datos con una distribución claramente diferente a $p_X(x)$. Si el **error cuadrático**[2] entre la imagen y la reconstrucción es grande para el caso del conjunto de

[2]Este error cuadrático no es un error cuadrático medio, porque se define para una sola muestra.



(a) (b) (c)

Figura 4.4: Ejemplo de detector de anomalías. Se verifica el funcionamiento de un autoencoder entrenado con imágenes de dígitos manuscritos frente a una clara anomalía. La imagen original fue extraída de `https://www.clipartmax.com/max/m2i8d3m2m2b1K9d3/`.

testeo anómalo y bajo para el conjunto de testeo original, se puede concluir que el autoencoder no aprendió una transformación biyectiva. Un ejemplo de este fenómeno puede verse en la Fig. 4.4, para un autoencoder entrenado con imágenes de dígitos manuscritos de 28×28 píxeles en escala de grises. Tomando una imagen con una distribución totalmente diferente, y adaptándola al formato de 28×28 en escala de grises, podemos ver que el autoencoder reconstruye una imagen fuertemente errónea. En el siguiente ejemplo se detallará como utilizar un autoencoder como detector de anomalías.

Ejemplo 4.3 *Se utiliza un autoencoder para implementar un detector de anomalías. Los errores cuadráticos de cada muestra fueron*

$$0.1, \quad 0.8, \quad 1.3, \quad 1.9, \quad 2.2$$

Sabiendo que las únicas muestras realmente anómalas fueron las de error 1.3 *y* 2.2*.*

- *Graficar la curva ROC. Indicar los puntos correspondientes a todos los umbrales posibles e interporlar dichos puntos con rectas.*
- *Indicar el EER y el AUC, basado en la interpolación anterior.*

Para un detector de anomalías, se define como detector $\varphi_t(x) = \mathbf{1}\{(x - \hat{x})^2 > t\}$ para $t > 0$: un clasificador que considerará anomalía a muestras con alto error cuadrático de reconstrucción. Para las 5 muestras dadas, serán pocos los valores de t que modifican realmente la clasificación. Para $t \in (-\infty; 0.1)$ todos los clasificadores serán iguales, lo mismo para $t \in [0.1, 0.8)$, para $t \in [0.8, 1.3)$, para $t \in [1.3, 1.9)$, para $t \in [1.9, 2.2)$ y para $t \in [2.0, \infty)$. En total la curva ROC estará formada por 6 puntos.

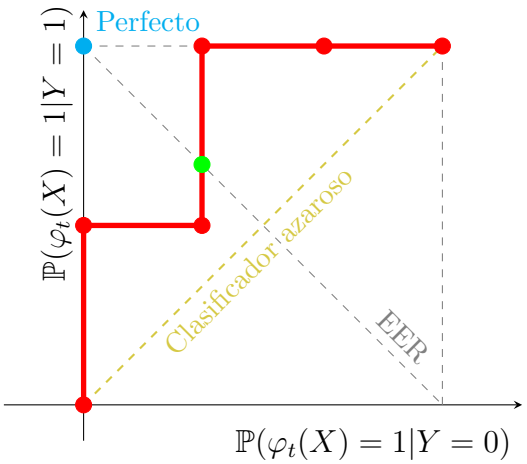


Figura 4.5: Curvas ROC para el detector de anomalías del Ej. 4.3.

En la Fig. 4.5 puede verse la curva ROC en cuestión. Cada umbral corresponde a un punto que será calculado de forma empírica, estudiando la proporción de muestras que cumplen la condición:

- Para $t \in (-\infty; 0.1)$, todas las muestras poseen un $\varphi_t(x) = 1$ y por lo tanto el punto de la ROC será el $(1, 1)$.
- Para $t \in [0.1, 0.8)$, todas las muestras anómalas $Y = 1$ son consideradas anómalas $\varphi_t(x) = 1$ y de las 3 muestras regulares $Y = 0$, dos serán consideradas anómalas. Por lo tanto el punto de la ROC es el $\left(\frac{2}{3}, 1\right)$.
- Para $t \in [0.8, 1.3)$, todas las muestras anómalas son consideradas anómalas y de las 3 muestras regulares, solo una será consideradas anómalas. Por lo tanto el punto de la ROC es el $\left(\frac{1}{3}, 1\right)$.
- Para $t \in [1.3, 1.9)$, una de dos anomalías será bien clasificada mientras que de las 3 muestras regulares solo una será consideradas anómalas. Por lo tanto el punto de la ROC es el $\left(\frac{1}{3}, \frac{1}{2}\right)$.
- Para $t \in [1.9, 2.2)$, todas las muestras regulares serán bien clasificadas, pero solo una de las dos anomalías será detectada. Por lo tanto el punto de la ROC es el $\left(0, \frac{1}{2}\right)$.
- Para $t \in [2.0, \infty)$, todas las muestras serán consideradas regulares y por lo tanto el punto de la ROC es el $(0, 0)$.

Gráficamente se puede observar que el EER se dará en la intersección de las rectas $\mathbb{P}(\varphi_t(X) = 1|Y = 1) = 1 - \mathbb{P}(\varphi_t(X) = 1|Y = 0)$ y $\mathbb{P}(\varphi_t(X) = 1|Y = 0) = \frac{1}{3}$. Es decir que $\text{EER} = \frac{1}{3}$. Para el caso del área basta con sumar áreas de rectángulos para notar que $\text{AUC} = \frac{5}{6}$.

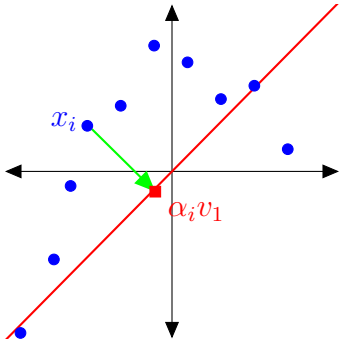


Figura 4.6: Ejemplo de reducción de dimensión (2D a 1D) utilizando PCA. En verde se destaca la proyección sobre el manifold de una de las muestras.

4.3. Análisis de Componentes Principales

El método lineal más utilizado en reducción de dimensión se denomina PCA (Análisis de Componentes Principales). PCA utiliza teoría de subespacios para proyectar ortogonalmente las muestras dentro del manifold hallado. Como todo subespacio, tiene la característica de pasar por el origen. Por este motivo es imprescindible normalizar en media a los datos[3] $(\tilde{x}_i)_j = (x_i)_j - \mu_j$ con $\mu_j = \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} (x_i)_j$. Ocasionalmente se puede normalizar también en varianza, pero solo se reserva para casos particulares donde se necesite interpretar cuantitativamente el manifold (y por lo tanto las magnitudes deben estar en las mismas unidades para ser comparables) o para solucionar algunos problemas numéricos (diferentes a los del gradiente descendente estudiados en la Sección 2.3.1, relacionados con el número de condición de la matriz involucrada).

Dado un conjunto de datos de media muestral nula $\{\tilde{x}_i\}_{i=1}^{n_{\text{tr}}}$, llamamos **dirección principal** v_1 al versor que minimiza el error cuadrático medio entre los datos $\tilde{x}_i$ y su proyección ortogonal $\alpha_i v_1$:

$$v_1 = \underset{\substack{v_1 \in \mathbb{R}^{d_x}: \\ \|v_1\|^2 = 1}}{\arg\min} \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} \|\tilde{x}_i - \alpha_i v_1\|^2 \quad \text{s.t.} \quad v_1^T(\tilde{x}_i - \alpha_i v_1) = 0 \quad \forall\, i = 1, \cdots, n_{\text{tr}} \tag{4.7}$$

donde α_i representa al valor de la muestra i-ésima en el espacio proyectado, y la condición $v_1^T(\tilde{x}_i - \alpha_i v_1) = 0$ induce la ortogonalidad entre el espacio a proyectar y el error de dicha proyección. La Fig. 4.6 muestra un ejemplo de dos dimensiones de PCA. Las muestras (indicadas con puntos azules) son proyectadas ortogonalmente sobre un espacio de dimensión reducida (la recta roja). Utilizando el hecho que v_1 posee norma unitaria, se puede deducir al valor de α_i de la condición de ortogonalidad como $\alpha_i = v_1^T \tilde{x}_i$. Esta condición excede a las muestras de entrenamiento, incluso en inferencia, para proyectar cualquier valor $\tilde{x}$ basta con computar $\alpha = v_1^T \tilde{x}$. Pero volviendo a la etapa de entrenamiento, el

[3] Por $(x_i)_j$ nos referimos a la componente j-ésima del vector x_i.

versor v_1 debe cumplir:

$$v_1 = \underset{\substack{v_1 \in \mathbb{R}^{d_x}: \\ \|v_1\|^2 = 1}}{\arg\min} \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} [\tilde{x}_i - (v_1^T \tilde{x}_i) v_1]^T [\tilde{x}_i - (v_1^T \tilde{x}_i) v_1] \tag{4.8}$$

$$= \underset{\substack{v_1 \in \mathbb{R}^{d_x}: \\ \|v_1\|^2 = 1}}{\arg\min} \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} \|\tilde{x}_i\|^2 - (v_1^T \tilde{x}_i)^2 \tag{4.9}$$

$$= \underset{\substack{v_1 \in \mathbb{R}^{d_x}: \\ \|v_1\|^2 = 1}}{\arg\max} \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} (v_1^T \tilde{x}_i)^2 \tag{4.10}$$

$$= \underset{\substack{v_1 \in \mathbb{R}^{d_x}: \\ \|v_1\|^2 = 1}}{\arg\max} v_1^T \left(\frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} \tilde{x}_i \tilde{x}_i^T \right) v_1 \tag{4.11}$$

La matriz $\Sigma = \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} \tilde{x}_i \tilde{x}_i^T$ es una covarianza muestral por ser muestras de media nula. Este tipo de problema de optimización con restricciones puede resolverse utilizando el multiplicador de Lagrange λ, definiendo la función:

$$J(v_1) = v_1^T \Sigma v_1 - \lambda (v_1^T v_1 - 1) \tag{4.12}$$

Para encontrar el óptimo basta con igualar a cero la primera derivada (gradiente) respecto a v_1 (para más información sobre derivadas respecto vectores/matrices ver [9]): $\nabla J(v_1) = 2\Sigma v_1 - 2\lambda v_1 = 0$, es decir $\Sigma v_1 = \lambda v_1$. Esta condición se cumple para todos los autovectores v_1 de la matriz Σ con el correspondiente autovalor asociado λ. Sea $(\lambda, v_1) \in \mathcal{V}$ el conjunto de pares autovalores-autoversores de la matriz Σ, la optimización de (4.8) es:

$$v_1 = \underset{(\lambda, v_1) \in \mathcal{V}}{\arg\max} \ \lambda \cdot \|v_1\|^2 = \underset{(\lambda, v_1) \in \mathcal{V}}{\arg\max} \ \lambda \tag{4.13}$$

Es decir que la dirección principal v_1 corresponde al autoversor asociado al autovalor más grande. Este procedimiento se puede repetir para encontrar el 2do, 3er, etc. componente principal. El resultado son el 2do, 3er, etc autovalor con su autovector como dirección.

A continuación se analizará como debe ser interpretada la etapa de inferencia en el contexto de autoencoders. Supongamos una reducción de dimensión de d_x a d_z para el espacio latente $\mathcal{Z} = \mathbb{R}^{d_z}$ con $d_z \leq d_x$, y sea $\mathbf{V} \in \mathbb{R}^{d_x \times d_z}$ la matriz formada con los d_z autovectores principales. El *encoder* queda definido por la operación $z = \mathbf{V}^T x$ mientras que el *decoder* hace lo propio con la operación $\hat{x} = \mathbf{V} z$; ambas transformaciones lineales. Por ser los autovectores ortonormales debe cumplirse que $\mathbf{V}^T \mathbf{V} = \mathbf{I}$, pero solo en el caso de que la matriz cuente con todos los autovectores (ya que la matriz $\mathbf{V}$ será inversible) será válido que $\mathbf{V}\mathbf{V}^T = \mathbf{I}$. El porcentaje de energía perdida en la reconstucción puede medirse por la proporción de autovalores despreciados (los autovalores son no negativos).

Edad	0-9	10-19	20-29	30-39	40-49	50-59	60-69	70-79	≥ 80	Total
Italia	0 % (0/43)	0 % (0/85)	0 % (0/296)	0 % (0/470)	0.1 % (1/891)	0.2 % (3/1453)	2.5 % (37/1471)	6.4 % (114/1785)	13.2 % (202/1532)	4.4 % (357/8026)
China	0 % (0/0)	0.2 % (1/549)	0.2 % (7/3619)	0.2 % (18/7600)	0.4 % (38/8571)	1.3 % (130/10008)	3.6 % (309/8583)	8 % (312/3918)	14.8 % (208/1408)	2.3 % (1023/44672)

Cuadro 4.1: Paradoja de Simpson presente sobre la tasa de letalidad del Covid-19 [25]. El total parecería indicar una letalidad mayor en Italia, sin embargo al desagregar la información por rango etario se observa que la tendencia era la contraria.

4.4. Clustering y el algoritmo K-Means

El problema de *clustering*, es decir codificar sobre un espacio finito $\mathcal{Z} = \{1, \cdots, K\}$, debe ser interpretado como el equivalente no supervisado de la clasificación. En el, a cada muestra x se le asigna una clase k como etapa de codificación. Para la decodificación, basta con reemplazar cada clase con un representante $\hat{x} \in \mathbb{R}^{d_x}$ de la misma, habitualmente llamado **centroide**. Al haber solamente K valores válidos de centroides, se vuelve imposible aprender un encoder biyectivo y por lo tanto es un problema menos a verificar.

En cuanto a aplicaciones, existen de lo más diversas: separar datos, comprimir información, caracterizar comportamientos estándar, entre otros. Pero uno de los más importantes es evitar la llamada **Paradoja de Simpson**. La paradoja de Simpson es un fenómeno estadístico en el cuál una tendencia que aparece en varios grupos de datos desaparece cuando estos grupos se combinan y en su lugar aparece la tendencia contraria para los datos agregados. Un ejemplo de este fenómeno puede verse en el Cuadro 4.1. En él, se observan los primeros datos de tasa de letalidad del Covid-19 en Italia y China. A primera vista, la información parecería indicar que el virus fue más letal en Italia (4.4 %) que en China (2.3 %). Sin embargo, cuando la información es desagregada por rango etario, se observa que consistentemente el Covid-19 fue más letal en China en todos los grupos. Lo que está pasando es que en Italia los pacientes morían porque su población era más longeva, pero esto no tenía que ver con las características del virus. Dividir la población en clases evitó llegar a conclusiones incorrectas.

El algoritmo más sencillo para resolver el problema de clustering se denomina **K-means**. Este algoritmo se basa en encontrar, de forma iterativa, los centroides de cada clase (como el centro de masa de la nube de puntos) y asignar cada muestra al centroide más cercano. Prefijados el número de clases K e inicializando los centroides eligiendo K muestras al azar del conjunto de datos de entrenamiento, *K-means* propone iterar hasta la convergencia entre:

- **Minimización (M):** Asignar a cada muestra x_i la clase k correspondiente al centroide μ_k más cercano $z_i = \arg \min_{k=1,\cdots,K} \|x_i - \mu_k\|$.
- **Expectación (E):** Recalcular cada centroide como el punto medio dentro de la clase correspondiente $\mu_k = \frac{1}{\#(z_i = k)} \sum_{i:\, z_i = k} x_i$.

Cabe destacar que una característica relevante de los algoritmos de aprendizaje no supervisado es su sensibilidad al punto de inicialización. En este tipo de algoritmos se suele necesitar correr el algoritmo muchas veces y analizar la consistencia de los resultados.

Ejemplo 4.4 *Clusterizar, via algoritmo K-means, el siguiente conjunto de datos. Incializar los centroides como* $(0, 3)$ *y* $(1, 1)$*. Indicar los resultados parciales de cada paso del algoritmo.*

idx	x_1	x_2
0	*0*	*3*
1	*1*	*1*
2	*2*	*0*
3	*3*	*2*

Cabe destacar que en este ejemplo $K = 2$ ya que se informan dos centroides.

- Como primer paso M, se asignará un valor $z_i = 1$ a las muestras más cercanas al $\mu_1 = (0, 3)$ y un valor $z_i = 2$ a las más cercanas al $\mu_2 = (1, 1)$. En este caso $z_0 = 1$, $z_1 = 2$, $z_2 = 2$ y $z_3 = 2$.
- Como primer paso E, se recalcularan los centroides $\mu_1 = (0, 3)$ y $\mu_2 = (2, 1)$.
- El 2do paso M vuelve a asignar una clase a cada muestra. Esta asignación vuelve a ser $z_0 = 1$, $z_1 = 2$, $z_2 = 2$ y $z_3 = 2$, denotando la convergencia alcanzada.

4.5. Algoritmo Expectación-Maximización

En problemas de aprendizaje no supervisado ambiciosos, donde se desea caracterizar totalmente la distribución de los datos, el estimador de máxima verosimilitud discutido en la Sección 1.2.3 es un buen punto de partida. Sin embargo, el problema con este tipo de algoritmos radica en que el estimador no tiene solución analítica para modelos medianamente sofisticados. Una solución numérica puede tampoco ser satisfactoria en grandes dimensiones, por volverse rápidamente un problema computacionalmente intratable. Es entonces cuando surge la necesidad de relajar el problema de optimización presente en este tipo de estimadores con un algoritmo iterativo.

Un primer detalle a tener en cuenta es el vínculo entre maximizar la verosimilitud y minimizar la entropía cruzada empírica. Dicho vínculo se evidencia utilizando la monotonía del logaritmo y la independencia entre muestras:

$$\arg\max_{\theta\in\Theta} p(\mathbf{x}|\theta) = \arg\max_{\theta\in\Theta} \prod_{i=1}^{n} p(x_i|\theta) = \arg\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} -\log p(x_i|\theta) \tag{4.14}$$

donde esta relación demuestra lo relevante que pueden ser las métricas de información (véase Sección 3.1.2.1) en este tipo de algoritmos. Entre estas métricas, vale la pena destacar a la divergencia de Kullback Leibler, la cuál cuantifica la discrepancia entre dos distribuciones[4]. Recordando que esta magnitud es siempre mayor o igual que cero con igualdad si y solo si las distribuciones son las mismas (Prop. 3.3), podemos reescribir al estimador como:

$$\hat{\theta}_{\text{MV}} = \arg\max_{\theta\in\Theta} \log p(\mathbf{x}|\theta) = \arg\max_{\theta\in\Theta} \max_{q\in\mathcal{P}} \log p(\mathbf{x}|\theta) - \text{KL}\left(q(\cdot|\mathbf{x})\|p(\cdot|\mathbf{x},\theta)\right) \tag{4.15}$$

donde $\mathcal{P}$ es la familia de todas las posibles distribuciones condicionales de un vector aleatorio $\mathbf{z}|\mathbf{x}$. La técnica descripta en la presente sección requiere una selección delicada de el vector $\mathbf{z}$ para cada problema. Una buena selección radicará en elegir una variable *no-observable* del problema relevante para el mismo, el conocimiento específico de la tarea a resolver es vital para este tipo de algoritmos. Este tipo de variables no observables, presentes en el algoritmo, harán las veces de **variables latentes**. A su vez se define la cota inferior esperada (*Expected Lower BOund*) como:

$$\text{ELBO}(\theta, q) = \log p(\mathbf{x}|\theta) - \text{KL}\left(q(\cdot|\mathbf{x})\|p(\cdot|\mathbf{x},\theta)\right) \tag{4.16}$$

donde su máximo alcanza la log-verosimilitud de la muestra (es una cota inferior).

El algoritmo *Expectación-Maximización* (o algoritmo EM), propone relajar el problema de optimización descripto en (4.15) resolviendo las dos maximizaciones de forma iterativa. Es decir, partiendo de un $\theta_0 \in \Theta$, el algoritmo propone iterar para $t \in \mathbb{N}$ entre los pasos de *expectación* y *maximización*:

$$E)\ q^{(t)} = \arg\max_{q\in\mathcal{P}} \text{ELBO}(\theta^{(t-1)}, q), \qquad M)\ \theta^{(t)} = \arg\max_{\theta\in\Theta} \text{ELBO}(\theta, q^{(t)}) \tag{4.17}$$

La solución de este algoritmo $\hat{\theta}$, en principio no tiene por que coincidir con el estimador de máxima verosimilitud $\hat{\theta}_{\text{MV}}$, por lo que será considerada *subóptima*. Sin embargo, en muchas familias paramétricas dicha solución será alcanzada, con habitual sensibilidad al punto de inicialización θ_0. En general, lo que siempre está asegurada es la monotonía de la verosimilitud. Es decir que en cada paso del algoritmo, la verosimilitud solo puede mejorar (aunque sin garantías de alcanzar $p(\mathbf{x}|\hat{\theta}_{\text{MV}})$).

[4]En rigor de verdad, en la Sección 3.1.2.1 fue definida la divergencia de Kullback Leibler para variables discretas. Sin embargo, ésta puede generalizarse de forma inmediata al caso continuo. Véase [2, Capítulo 8] para más detalles.

Propiedades 4.1 $p(\mathbf{x}|\theta^{(t)}) \geq p(\mathbf{x}|\theta^{(t-1)})$.

Demostración 4.4 (Prop. 4.1) *El algoritmo EM maximizaciones sucesivas, logrando así una cadena acendente de ELBOs*

$$ELBO\left(\theta^{(t-1)}, q^{(t)}\right) \leq ELBO\left(\theta^{(t)}, q^{(t)}\right) \leq ELBO\left(\theta^{(t)}, q^{(t+1)}\right) \tag{4.18}$$

En particular, el paso E se puede analizar estudiando la Prop. 3.3 aplicada a (4.16) *(elegir el* $q^{(t)}$ *que haga cero la divergencia de Kullback Leibler). Esto nos lleva a identidades del tipo* $p(\mathbf{x}|\theta^{(t-1)}) = ELBO\left(\theta^{(t-1)}, q^{(t)}\right)$ *y* $p(\mathbf{x}|\theta^{(t)}) = ELBO\left(\theta^{(t)}, q^{(t+1)}\right)$. *Reemplazando estas identidades en* (4.18) *la propiedad es demostrada.*

El paso M o *maximización*, recibe su nombre por representar una maximización numérica (igualando a cero la derivada de alguna expresión). Su expresión puede simplificarse utilizando la siguiente cadena de razonamientos:

$$\text{ELBO}(\theta, q) = \mathbb{E}_q\left[p(\mathbf{x}|\theta) - \log\frac{q(\mathbf{Z}|\mathbf{x})}{p(\mathbf{Z}|\mathbf{x},\theta)}\middle|\,\mathbf{X}=\mathbf{x}\right] \tag{4.19}$$

$$= \mathbb{E}_q\left[\log p(\mathbf{x},\mathbf{Z}|\theta)|\mathbf{X}=\mathbf{x}\right] - \mathbb{E}_q\left[\log q(\mathbf{Z}|\mathbf{x})|\mathbf{X}=\mathbf{x}\right] \tag{4.20}$$

donde el término restante no depende de θ y por lo tanto no influirá en este paso. Es decir, el paso M se basa en resolver

$$\frac{\partial}{\partial\theta}\mathbb{E}_{q^{(t)}}\left[\log p(\mathbf{x},\mathbf{Z}|\theta)|\mathbf{X}=\mathbf{x}\right] = 0 \tag{4.21}$$

Por su parte, el paso E o *expectación*, simplemente elige $q^{(t)}(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x},\theta^{(t-1)})$ para volver cero la divergenica de Kullback Leibler de (4.16). El mismo puede interpretarse como un *proxy* para el cómputo de $\mathbb{E}_{q^{(t)}}\left[\log p(\mathbf{x},\mathbf{Z}|\theta)|\mathbf{X}=\mathbf{x}\right]$, magnitud necesaria en el próximo paso M. Dado que su función puede reducirse a calcular una esperanza es que recibe el nombre de *expectación* (spanglish). En resumen, el algoritmo EM puede simplificarse como:

$$E)\ q^{(t)}(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x},\theta^{(t-1)}), \qquad M)\ \left.\frac{\partial}{\partial\theta}\mathbb{E}_{q^{(t)}}\left[\log p(\mathbf{x},\mathbf{Z}|\theta)|\mathbf{X}=\mathbf{x}\right]\right|_{\theta=\theta^{(t)}} = 0 \tag{4.22}$$

A continuación se analizará un ejemplo para fijar conceptos.

Ejemplo 4.5 *Los habitantes de* Smallville *pueden ser considerados* trabajador registrado, trabajador informal *o* desempleado *con probabilidades* $\frac{\theta}{2}$, $\frac{1-\theta}{2}$ *y* $\frac{1}{2}$ *respectivamente, donde* $0 \leq \theta \leq 1$. *El municipio posee* 10.000 *habitantes y cuenta con* 4.000 *trabajadores registrados.*

1. *Estimar* θ *por máxima verosimilitud.*
2. *Deducir matemáticamente una recursión, vía algoritmo EM, que permita estimar* θ.

3. *Se denomina* puntos fijos *a los valores de θ que no varían al iterar un paso del algoritmo. Encontrar los puntos fijos del problema de recursión definido por EM.*

4. *¿A que punto converge el problema si $\theta_0 = 0.999$? Analizar resultado.*

Sea X la cantidad de trabajadores registrados, Y la cantidad de trabajadores informales y Z la cantidad de desempleados.

1. La única variable observable es $x = 4000$, ya que con los datos públicos uno no puede distinguir con precisión entre informales y desempleados. En este caso, la probabilidad estimada por máxima verosimilitud es $\frac{4000}{10000} = \frac{2}{5}$. A su vez, el principio de invarianza nos dice que si dicha probabilidad es modelada como $\frac{\theta}{2}$, entonces $\hat{\theta}_{\text{MV}} = \frac{4}{5}$.

2. Por la naturaleza del modelo, las variables aleatorias involucradas forman un vector con distribución $(X, Y, Z) \sim \mathcal{M}_n(10000, \left[\frac{\theta}{2}, \frac{1-\theta}{2}, \frac{1}{2}\right])$ (véase Def. 1.2). Podemos elegir como variable latente, tanto la cantidad de trabajadores informales como la cantidad de desempleados, ya que ambas son variables relevantes del problema. En particular, elegimos Z para mantener la nomenclatura. Luego $Z|X = 4000 \sim \text{Bin}(6000, \frac{1}{2-\theta})$, definiendo $p(z|x, \theta)$ y por lo tanto el paso E. La distribución es binomial por dejar definido experimentos del tipo éxito/fracaso (es o no desempleado), la cantidad de experimentos (potenciales desempleados) son 6000 ya que 4000 de los 10000 ciudadanos son sabidos formales, y la probabilidad fue calculada como ser desempleado sabiendo que no es formal:

$$\mathbb{P}(\text{desempleado}|\text{no formal}) = \frac{\mathbb{P}(\text{desempleado})}{1 - \mathbb{P}(\text{formal})} = \frac{\frac{1}{2}}{1 - \frac{\theta}{2}} = \frac{1}{2 - \theta} \tag{4.23}$$

Por el lado del paso M, se puede utilizar la función de la probabilidad de la multinomial como[5]:

$$\log P(x, z|\theta) = k_{x,z} + x \log \frac{\theta}{2} + (10000 - x - z) \log \frac{1 - \theta}{2} \tag{4.24}$$

donde $k_{x,z}$ acapara constantes que no dependen de θ. Luego,

$$\mathbb{E}_{Q^{(t)}} \left[\log P(4000, Z|\theta)|X = 4000\right] = k + 4000 \log \frac{\theta}{2} + (6000 - \mathbb{E}_{Q^{(t)}} \left[Z|X = 4000\right]) \log \frac{1 - \theta}{2} \tag{4.25}$$

donde k acapara constantes que no dependen de θ. El algoritmo queda definido con un paso E que define $Q^{(t)}$ a partir de $\theta^{(t-1)}$ computando $\mathbb{E}_{Q^{(t)}} \left[Z|X = 4000\right] = \frac{6000}{2-\theta^{(t-1)}}$, y un paso M que define $\theta^{(t)}$ a partir de $Q^{(t)}$ igualando a cero la derivada de

[5]Utilizaremos P y Q mayúsculas, en lugar de p y q, para resaltar que las variables involucradas son discretas.

la expresión (4.25):

$$\frac{4000}{\theta} - \frac{6000 - \mathbb{E}_{Q^{(t)}}\left[Z|X = 4000\right]}{1-\theta} = 0 \tag{4.26}$$

Las siguientes recursiones son soluciones equivalentes:

$$\frac{4000}{\theta^{(t)}} = \frac{6000 - \frac{6000}{2-\theta^{(t-1)}}}{1-\theta^{(t)}} \tag{4.27}$$

$$2(1-\theta^{(t)}) = \left(3 - \frac{3}{2-\theta^{(t-1)}}\right)\theta^{(t)} \tag{4.28}$$

$$2 = \left(5 - \frac{3}{2-\theta^{(t-1)}}\right)\theta^{(t)} \tag{4.29}$$

y por lo tanto

$$\theta^{(t)} = \frac{2}{5 - \frac{3}{2-\theta^{(t-1)}}} = \frac{4 - 2\theta^{(t-1)}}{7 - 5\theta^{(t-1)}} \tag{4.30}$$

3. Para los puntos fijos basta con buscar los valores de $\theta = \theta^{(t)} = \theta^{(t-1)}$. Para ellos puede despejarse $5\theta^2 - 9\theta + 4 = 0$. Esta parábola tiene soluciones $\theta = \frac{4}{5}$ y $\theta = 1$.

4. Los puntos fijos indican los únicos valores a los que puede converger el algoritmo. En este caso hay dos: el estimador de máxima verosimilitud $\theta = \hat{\theta}_{\text{MV}}$ y la no existencia de trabajadores informales $\theta = 1$. Mientras que una es la solución que buscamos, la otra contradice el modelo propuesto. Sin embargo, la naturaleza de ambas es muy diferente. Mientras que $\theta = \hat{\theta}_{\text{MV}}$ es una solución atractiva, $\theta = 1$ es repulsiva. Queda al lector probar que incluso inicializando muy cerca de $\theta = 1$ ($\theta_0 = 0.999$), el algoritmo converge a $\theta = \hat{\theta}_{\text{MV}}$.

El ejemplo anterior funciona a modo pedagógico, obviamente cuando es sencillo calcular el estimador de máxima verosimilitud no tiene sentido plantear el algoritmo EM. A continuación se presentan algunos de los modelos más habituales en el uso de este algoritmo.

4.5.1. Modelo de Mezclas

Los modelos más habituales en el uso del algoritmo EM son los que caracterizan a las muestras x_i como mezclas. En ese contexto, cada variable x_i posee su correspondiente variable mezcladora z_i. Sea una muestra $n-$dimensional de pares independientes (X, Z), con $Z \sim \text{Cat}(\{c_1, \cdots, c_K\})$ y $\theta = \{c_k, \theta_k\}_{k=1}^K$, donde los θ_k representan los parámetros de la distribución $X|Z = k$. Este tipo de modelos se denomina *soft-clustering*, porque no solamente se asocia una etiqueta para cada x, sino que se caracteriza la probabilidad de pertenecer a cada cluster a partir de $Q(k|x)$.

En cuanto a su implementación, notar que para el paso E basta con utilizar la independencia entre las muestras $Q^{(t)}(\mathbf{z}|\mathbf{x}) = \prod_{i=1}^n P(z_i|x_i, \theta^{(t-1)})$ (la variable k indexa los valores

de $\mathcal{Z}$). Es decir que $Q^{(t)}(k|x) = P(k|x, \theta^{(t-1)})$, donde

$$Q^{(t)}(k|x) = \frac{c_k^{(t-1)} \cdot p(x|Z_i = k, \theta_k^{(t-1)})}{\sum_{m=1}^{K} c_m^{(t-1)} \cdot p(x|Z_i = m, \theta_m^{(t-1)})} \tag{4.31}$$

Con esta distribución se calcula la esperanza necesaria para el paso M

$$\mathbb{E}_{Q^{(t)}}\left[\log p(\mathbf{x}, \mathbf{Z}|\theta)|\mathbf{X} = \mathbf{x}\right] = \sum_{i=1}^{n} \mathbb{E}_{Q^{(t)}}\left[\log p(Z_i|\theta) + \log p(x_i|Z_i, \theta)|X_i = x_i\right] \tag{4.32}$$

$$= \sum_{i=1}^{n}\sum_{k=1}^{K} Q^{(t)}(k|x_i)\left[\log c_k + \log p(x_i|Z_i = k, \theta_k)\right] \tag{4.33}$$

con la restricción de $\sum_{k=1}^{K} c_k = 1$. Dicha restricción puede incorporarse al problema utilizando *multiplicadores de Lagrange*. Es decir, que en el paso M se elegirá como $\theta^{(t)}$ los parámetros que igualen a cero la derivada de:

$$J(\theta) = \sum_{i=1}^{n}\sum_{k=1}^{K} Q^{(t)}(k|x_i)\left[\log c_k + \log p(x_i|Z_i = k, \theta_k)\right] + \lambda\left(1 - \sum_{k=1}^{K} c_k\right) \tag{4.34}$$

Esta expresión se deberá derivar respecto a cada c_k y cada θ_k. Por el lado de c_k se obtiene:

$$\frac{\partial J(\theta)}{\partial c_k} = \sum_{i=1}^{n} \frac{Q^{(t)}(k|x_i)}{c_k} - \lambda = 0 \;\rightarrow\; c_k = \frac{1}{\lambda}\sum_{i=1}^{n} Q^{(t)}(k|x_i) \tag{4.35}$$

Notar que, para que $\sum_{k=1}^{K} c_k = 1$, es necesario que $\lambda = n$, obteniendo entonces la condición:

$$c_k^{(t)} = \frac{1}{n}\sum_{i=1}^{n} Q^{(t)}(k|x_i) \tag{4.36}$$

Por el lado de los parámetros θ_k se deberá resolver:

$$\frac{\partial J(\theta)}{\partial \theta_k} = \sum_{i=1}^{n} Q^{(t)}(k|x_i)\frac{\partial \log p(x_i|Z_i = k, \theta_k)}{\partial \theta_k} = 0 \tag{4.37}$$

Mientras que (4.31) y (4.36) aplican para cualquier modelo de mezcla, (4.37) deberá adaptarse a cada modelo particular. En este contexto, el más habitual se conoce como *mezcla de Gaussianas*.

4.5.1.1. Modelo de mezcla de Gaussianas

El modelo más utilizado por este algoritmo es el de mezcla de gaussianas. En él, se utilizan distribuciones normales multivariadas de la forma $X|Z = k \sim \mathcal{N}(\mu_k, \Sigma_k)$, donde $\theta_k = (\mu_k, \Sigma_k)$. Es decir, que (4.37) se traduce en dos conjuntos de parámetros a actualizar: μ_k y Σ_k. En este caso, es fácil notar que

$$\log p(x|Z = k, \theta_k) = \text{cte} - \frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x - \mu_k)^T\Sigma_k^{-1}(x - \mu_k) \tag{4.38}$$

donde cte es un término que no depende de θ_k (y por lo tanto se eliminará al calcular la derivada). Al derivar respecto de la media se obtiene (para detalles acerca de la derivada respecto a vectores y matrices ver [9]):

$$\frac{\partial \log p(x|Z=k,\theta_k)}{\partial \mu_k} = \Sigma_k^{-1}(x-\mu_k) \tag{4.39}$$

por lo tanto, la actualización de la media se deberá efectuar con la siguiente ecuación:

$$\sum_{i=1}^{n} Q^{(t)}(k|x_i)\Sigma_k^{-1}(x_i-\mu_k) = 0 \;\rightarrow\; \mu_k^{(t)} = \frac{\sum_{i=1}^{n} Q^{(t)}(k|x_i)\cdot x_i}{\sum_{i=1}^{n} Q^{(t)}(k|x_i)} \tag{4.40}$$

En cambio, al derivar respecto de la covarianza se obtiene:

$$\frac{\partial \log p(x|Z=k,\theta_k)}{\partial \Sigma_k} = -\frac{1}{2}\Sigma_k^{-1} + \frac{1}{2}\Sigma_k^{-1}(x-\mu_k)(x-\mu_k)^T\Sigma_k^{-1} \tag{4.41}$$

Esto quiere decir que, para la covarianza, (4.37) se traduce en:

$$\sum_{i=1}^{n} Q^{(t)}(k|x_i)\left[-\frac{1}{2}\Sigma_k^{-1} + \frac{1}{2}\Sigma_k^{-1}(x_i-\mu_k)(x_i-\mu_k)^T\Sigma_k^{-1}\right] = 0 \tag{4.42}$$

con lo cuál, la actualización de la covarianza deberá ser:

$$\Sigma_k^{(t)} = \frac{\sum_{i=1}^{n} Q^{(t)}(k|x_i)\cdot(x_i-\mu_k^{(t)})(x_i-\mu_k^{(t)})^T}{\sum_{i=1}^{n} Q^{(t)}(k|x_i)} \tag{4.43}$$

Es decir, que el algoritmo EM queda definido por las ecuaciones (4.31), (4.36), (4.40) y (4.43).

4.5.2. Análisis de Factores

El uso de variables latentes permite plantear el problema de reducción de dimensión (véase Sec. 4.2), desde una perspectiva más general, como un problema estocástico. Se modelan los datos observados como:

$$X = \mu + W\cdot Z + \epsilon \tag{4.44}$$

donde $\mu \in \mathbb{R}^{d_x}$, $W \in \mathbb{R}^{d_x \times d_z}$, $Z \sim \mathcal{N}(0, I)$ (de dimensión d_z) y $\epsilon \sim \mathcal{N}(0, \Psi)$ (de dimensión d_x) con $\Psi \in \mathbb{R}^{d_x \times d_x}$ una matriz diagonal y con Z y ϵ independientes. En particular, si se restringe el problema a una covarianza esférica $\Psi = \sigma^2\mathbf{I}$, se obtiene un algoritmo conocido como Probabilistic PCA.

Este modelo separa los datos como una suma de factores: la media μ, un factor representativo del *manifold* WZ y un ruido ϵ independiente de todo lo demás. Combinando la normalidad de las variables aleatorias con la aditividad de los factores, queda asegurada tanto la normalidad conjunta entre (X, Z) como condicional $Z|X = x$ y $X|Z = z$, lo cuál simplifica bastante los cálculos del modelo. Para este problema, los parámetros son

$\{\mu, W, \Psi\}$ y serán aprendidos por el algoritmo EM.

Esta caracterización estocástica, permite plantear el problema de reducción de dimensión, utilizando como encoder como $\mathbb{E}[Z|X = x]$ y como decoder $\mathbb{E}[X|Z = z]$ (las esperanzas condicionales minimizan el error cuadrático medio como se demostró en la Prop. 1.8). Este modelo puede ser utilizado como una variante de PCA (véase Sec. 4.3), también lineal, con el agregado de funcionar también como método generativo: pasando por el decoder a variables latentes generadas como $Z \sim \mathcal{N}(0, I)$, es factible generar nuevas muestras.

La etapa de predicción (testeo, validación, etc.) quedará definido por las expresiones $\mathbb{E}[Z|X = x]$ y como decoder $\mathbb{E}[X|Z = z]$, donde el decoder se puede ver inmediatamente de (4.44) que $X|Z = z \sim \mathcal{N}(\mu + Wz, \Psi)$ y por lo tanto $\mathbb{E}[X|Z = z] = \mu + Wz$. El encoder en cambio será calculado durante el paso E: $q^{(t)}(\mathbf{z}|\mathbf{x}) = \prod_{i=1}^{n} p(z_i|x_i, \theta^{(t-1)})$, es decir que $q^{(t)}(z|x) = p(z|x, \theta^{(t-1)})$.

En cuanto a su entrenamiento, se descomponerá el los previamente pasos definidos E y M. En cuanto al paso E, se desea caracterizar los parámetros de la distribución de $Z|X = x$ (la cuál será normal). En primer lugar, la distribución $Z \sim \mathcal{N}(0, I)$ es definida por el modelo, mientras que $X \sim \mathcal{N}(\mu, WW^T + \Psi)$ (utilizando la independencia entre Z y ϵ, y las propiedades vectoriales de la covarianza [9][6]). Entonces, la conjunta posee distribución:

$$\begin{bmatrix} X \\ Z \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ 0 \end{bmatrix}, \begin{bmatrix} WW^T + \Psi & W \\ W^T & I \end{bmatrix} \right) \tag{4.45}$$

donde la covarianza cruzada es consecuencia de

$$\mathbb{E}[(X - \mu)(Z - 0)^T] = \mathbb{E}[(WZ - \epsilon)Z^T] = W \tag{4.46}$$

Una vez reconocida la distribución conjunta como normal multivariaza, la condicional puede deducirse con propiedades de esta familia de distribuciones [9]:

$$Z|X = x \sim \mathcal{N}\left(W^T(WW^T + \Psi)^{-1}(x - \mu), I - W^T(WW^T + \Psi)^{-1}W\right) \tag{4.47}$$

Dos importantes características a resaltar de (4.47) son la linealidad del encoder $\mathbb{E}[Z|X = x] = W^T(WW^T + \Psi)^{-1}(x - \mu)$ y la *heterocedasticidad* (covarianza constante, no depende de x), definiendo $\Sigma = I - W^T(WW^T + \Psi)^{-1}W$. En este caso llamamos también $m_i = W^T(WW^T + \Psi)^{-1}(x_i - \mu)$ para las muestras de entrenamiento.

Las expresiones de m_i y Σ requieren el cómputo de una inversa de una matriz de dimensión d_x. Dado que en el problema de reducción de dimensión, $d_z < d_x$, suele aliviar el cómputo reescribiendo las expresiones utilizando propiedades de las inversas matriciales

[6]Es esta sección se utilizarán diversas propiedades matriciales. Se recomienda al lector seguir las deducciones consultando simultáneamente este apunte y el libro [9].

[9]:

$$m_i = \Sigma W^T \Psi^{-1}(x_i - \mu), \qquad \Sigma = (I + W^T \Psi^{-1} W)^{-1} \tag{4.48}$$

En cuanto al paso M hay varias consideraciones a tener en cuenta. En primer lugar, para simplificar el análisis se aplicará EM solamente sobre $\theta = \{W, \Psi\}$ y se estimará la media utilizando la media muestral $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ (la cuál será calculada previa a la iteración de EM por no necesitar actualizarse). Esto no es necesario de imponerlo para llegar a los mismos resultados, asumiendo solamente $\mu^{(0)} = \frac{1}{n}\sum_{i=1}^{n} x_i$ y buscando su recursión por EM se demuestra el mismo resultado (será demostrado al final del capítulo). Pero el excluirlo de antemano facilita el análisis. En segundo lugar, una vez establecido el valor de la media, queda de manifiesto que $\sum_{i=1}^{n} m_i = 0$.

En tercer lugar, el modelo propuesto, no solamente asume la independencia entre las muestras, sino que también impone una $p(z)$ independiente de los parámetros (normal estándar). Esto permite reescribir la esperanza necesaria para el paso M como:

$$\mathbb{E}_{q^{(t)}}\left[\log p(\mathbf{x}, \mathbf{Z}|\theta)|\mathbf{X} = \mathbf{x}\right] = \sum_{i=1}^{n} \mathbb{E}_{q^{(t)}}\left[\log p(Z_i)\right] + \mathbb{E}_{q^{(t)}}\left[\log p(x_i|Z_i, \theta)|X_i = x_i\right] \tag{4.49}$$

donde la primera esperanza no depende del parámetro (su derivada será nula) y la segunda es de la forma

$$\begin{aligned}&\mathbb{E}_{q^{(t)}}\left[\log p(x|Z, \theta)|X = x\right] = \\ &\quad -\frac{d_x}{2}\log(2\pi) - \frac{1}{2}\log|\Psi| - \frac{1}{2}\mathbb{E}_{q^{(t)}}\left[(x - \mu - WZ)^T \Psi^{-1}(x - \mu - WZ)\big| X = x\right]\end{aligned} \tag{4.50}$$

En cuarto lugar cabe destacar que, bajo $q^{(t)}(\cdot|x_i)$, $U = x_i - \mu - WZ \sim \mathcal{N}(x_i - \mu - Wm_i, W\Sigma W^T)$. Luego, la esperanza anterior puedo escribirla como [9]:

$$\mathbb{E}_{q^{(t)}}[U^T \Psi^{-1} U|X_i = x_i] = \mathrm{Tr}(\Psi^{-1} W\Sigma W^T) + (x_i - \mu - Wm_i)^T \Psi^{-1}(x_i - \mu - Wm_i) \tag{4.51}$$

Teniendo todas estas cuestiones en cuenta se puede definir a la función cuya derivada será igualada a cero:

$$J(W, \Psi) = \mathrm{cte} - \frac{n}{2}\log|\Psi| - \frac{n}{2}\mathrm{Tr}(\Psi^{-1} W\Sigma W^T) - \frac{1}{2}\sum_{i=1}^{n}(x_i - \mu - Wm_i)^T \Psi^{-1}(x_i - \mu - Wm_i) \tag{4.52}$$

Derivando respecto de W y utilizando propiedades matriciales [9]:

$$\frac{\partial J(W, \Psi)}{\partial W} = -n\Psi^{-1} W\Sigma - \frac{1}{2}\sum_{i=1}^{n}\left[2\Psi^{-1} Wm_i m_i^T - 2\Psi^{-1}(x_i - \mu)m_i^T\right] \tag{4.53}$$

Igualando a cero y despejando W:

$$W = \left(\sum_{i=1}^{n}(x_i - \mu)m_i^T\right)\left(\sum_{i=1}^{n} m_i m_i^T + \Sigma\right)^{-1} \tag{4.54}$$

$$= \left(\sum_{i=1}^{n} x_i m_i^T\right)\left(\sum_{i=1}^{n} m_i m_i^T + \Sigma\right)^{-1} \tag{4.55}$$

A continuación se computará la derivada respecto de Ψ y se utilizarán propiedades matriciales [9]:

$$\frac{\partial J(W,\Psi)}{\partial \Psi} = -\frac{n}{2}\Psi^{-1} + \frac{n}{2}\Psi^{-1}W\Sigma W^T\Psi^{-1} + \frac{1}{2}\sum_{i=1}^{n}\Psi^{-1}(x_i-\mu-Wm_i)(x_i-\mu-Wm_i)^T\Psi^{-1} \tag{4.56}$$

Igualando a cero y despejando Ψ:

$$\Psi = \frac{1}{n}\sum_{i=1}^{n} W\Sigma W^T + (x_i-\mu-Wm_i)(x_i-\mu-Wm_i)^T \tag{4.57}$$

$$= \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)(x_i-\mu)^T - Wm_i(x_i-\mu)^T - (x_i-\mu)m_i^TW^T + W(m_im_i^T+\Sigma)W^T \tag{4.58}$$

Utilizando la expresión anteriormente encontrada de W:

$$\Psi = \frac{1}{n}\left[\sum_{i=1}^{n}(x_i-\mu)(x_i-\mu)^T - W\sum_{i=1}^{n} m_i x_i^T\right] \tag{4.59}$$

Cabe aclarar que, dado que nuestro modelo impone un Ψ diagonal, nos terminamos quedando con la diagonal de la matriz encontrada (solo importa derivar con respecto a los elementos de la diagonal). En definitiva, el algoritmo EM en este caso propone iterar entre un paso E (4.48) y un paso M definido por las expresiones (4.55) y (4.59).

Demostración 4.5 (Incorporar μ en el algoritmo EM) *Se desea demostrar que al incorporar μ a la recursión EM, con condición inicial $\mu^{(0)} = \frac{1}{n}\sum_{i=1}^{n} x_i$, $\mu^{(t)} = \mu^{(0)}$ para todo $t \in \mathbb{N}$. Como las recursiones de EM solo dependen del paso anterior, es suficiente con mostrar que $\mu^{(1)} = \mu^{(0)}$ (el resto de la demostración es inducción). Esto es válido siempre y cuando se verifique que su valor tampoco depende del resto de parámetros que si se va actualizando.*

Es importante notar que la expresión (4.48) *asegura, al menos, $\sum_{i=1}^{n} m_i^{(1)} = 0$. Redefiniendo la función a minimizar teniendo en cuenta la media*

$$J(\mu, W, \Psi) = g(W,\Psi) - \frac{1}{2}\sum_{i=1}^{n}(x_i-\mu-Wm_i)^T\Psi^{-1}(x_i-\mu-Wm_i) \tag{4.60}$$

y derivando respecto de μ se obtiene

$$\frac{\partial J(\mu,W,\Psi)}{\partial \mu} = -\sum_{i=1}^{n}\Psi^{-1}(x_i-\mu-Wm_i) \tag{4.61}$$

Por lo tanto, igualando a cero esta derivada puede observarse que:

$$\mu^{(1)} = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - W^{(1)}m_i^{(1)}\right) = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.62}$$

5

Procesamiento de Datos orientado a Aplicaciones Específicas

El algoritmo no manda, castiga ni interroga, sino que sugiere, optimiza y predice. Esa suavidad operativa no lo vuelve menos violento, sino que más eficaz, porque captura al sujeto desde sus deseos y lo vuelve cómplice de su propia subordinación.

TODO

5.1. Procesamiento de Audio

5.1.1. Espectrograma

5.1.2. Coeficientes Mel-Cepstrum

5.2. Procesamiento de Texto

5.3. Sistemas de Recomendación

5.4. Ingeniería de Características

5.4.1. Test de Independencia Chi-Cuadrado

5.4.2. Tests ANOVA

Comparación medias de normales varianza conocida Comparación medias de normales varianza desconocida Coparación de varianza de normales Asintótico

6

Modelos Bayesianos

Es tan cierto que la inteligencia artificial solamente reproduce datos, como que los datos lo reproducen todo.
Ley de los Grandes Números.

Tras una implementación particularmente difícil de un algoritmo, los programadores suelen desconfiar de cu código y probarlo con un ejemplo trivial. Si lo supera comienzan a probar dicho código con ejemplos más y más complejos. Cuando el código ya superó cuatro o cinco pruebas de éstas, el programador empieza a creer que posiblemente no haya errores en ese código. En esto consiste el pensamiento bayesiano: actualizar las creencias tras considerar nueva evidencia [6].

6.1. Inferencia Bayesiana

La estadística bayesiana, discutida en la Sección 1.2.4, posee características particulares. Su filosofía radica en buscar verdades en contextos de incertidumbre, modelando no solo el problema a resolver sino también nuestra ignorancia sobre el mismo [7]. Algunas de las características técnicas son:

- Sea T una variable aleatoria representativa de los parámetros y las variables no observables del modelo, con distribución a priori $p_T(\theta)$.
- La estadística bayesiana supone una relación causal $T \to \mathbf{X}$, donde $\mathbf{X}$ es cualquier conjunto aleatorio de muestras a observar.
- La relación anterior implica la independencia entre las muestras *cuando se conoce el parámetro*. Es decir que la verosimilitud de una muestra puede escribirse como $p_{\mathbf{X}|T=\theta}(\mathbf{x}) = \prod_{i=1}^{n} p_{X|T=\theta}(x_i)$.
- La *distribución a posteriori* es proporcional al producto de la *prior* y la verosimilitud

$$p_{T|\mathbf{X}=\mathbf{x}}(\theta) \propto p_T(\theta) \cdot \prod_{i=1}^{n} p_{X|T=\theta}(x_i) \tag{6.1}$$

- Por último, se define la distribución predictiva bayesiana como:

$$p_{X_{\text{test}}|\mathbf{X}=\mathbf{x}}(x_{\text{test}}) = \int_{\Theta} p_{X|T=\theta}(x_{\text{test}}) p_{T|\mathbf{X}=\mathbf{x}}(\theta) d\theta = \mathbb{E}\left[p(x_{\text{test}}|T)|\mathbf{X}=\mathbf{x}\right] \tag{6.2}$$

  donde X_{test} es una variable aleatoria no vista en el conjunto de entrenamiento $\mathbf{X}$.

A continuación se presenta un ejemplo de como calcular analíticamente este tipo de distribuciones.

Ejemplo 6.1 *Lucas dispara a un blanco y el disparo impacta en un punto aleatorio* $(X, 0)$ *con* X *(en decímetros) una variable aleatoria con distribución normal de media nula y varianza* $1/\tau$*, donde* τ *representa la precisión de Lucas. A priori la precisión* τ *tiene una distribución chi-cuadrado de* 8 *grados de libertad. Lucas tiro* 10 *veces al blanco y observó* $\sum_{i=1}^{10} x_i^2 = 17$*. En virtud a la información muestral,*

1. *Hallar la distribución* a posteriori.
2. *Hallar la distribución predictiva.*

Como primer paso en un problema bayesiano, hay que comenzar planteando la distribución *a posteriori*. En este caso evitaremos las constantes de proporcionalidad:

$$p_{T|\mathbf{X}=\mathbf{x}}(\tau) \propto p_T(\tau) \cdot \prod_{i=1}^{n} p_{X|T=\tau}(x_i) \propto \tau^3 e^{-\tau/2} \mathbf{1}\{\tau > 0\} \cdot \prod_{i=1}^{10} \sqrt{\tau} e^{-\frac{\tau}{2} x_i^2} \tag{6.3}$$

donde se utilizó la información de que la versoimilitud es normal y la distribución *a priori* es chi-cuadrado (véase Cuadro 1.2). Utilizando el dato muestreal del enunciado, podemos observar que $p_{T|\mathbf{X}=\mathbf{x}}(\tau) \propto \tau^8 e^{-9\tau} \mathbf{1}\{\tau > 0\}$, es decir que la variable se distribuye *a posteriori* como $T|_{\mathbf{X}=\mathbf{x}} \sim \Gamma(9, 9)$ (véase Cuadro 1.2). La distribución predictiva es de la forma

$$p_{X_{\text{test}}|\mathbf{X}=\mathbf{x}}(x_{\text{test}}) = \int_{\Theta} p_{X|T=\tau}(x_{\text{test}}) p_{T|\mathbf{X}=\mathbf{x}}(\tau) d\tau \propto \int_0^{\infty} \sqrt{\tau} e^{-\frac{\tau}{2} x_{\text{test}}^2} \cdot \tau^8 e^{-9\tau} d\tau \tag{6.4}$$

Reconociendo el núcleo de la integral, se puede observar que el mismo es proporcional a la densidad de una $\Gamma(\nu, \beta)$. Sabiendo que por ser densidad debe integrar 1:

$$p_{X_{\text{test}}|\mathbf{X}=\mathbf{x}}(x_{\text{test}}) \propto \int_0^{\infty} \tau^{\frac{17}{2}} e^{-\tau\left(9+\frac{x_{\text{test}}^2}{2}\right)} d\tau \propto \left(9 + \frac{x_{\text{test}}^2}{2}\right)^{-\frac{19}{2}} \tag{6.5}$$

donde se utilizó $\nu = \frac{19}{2}$ y $\lambda = 9 + \frac{x_{\text{test}}^2}{2}$. Es decir que la densidad predictiva es proporcional a $p_{X_{\text{test}}|\mathbf{X}=\mathbf{x}}(x_{\text{test}}) \propto \left(1 + \frac{x_{\text{test}}^2}{18}\right)^{-\frac{18+1}{2}}$, y por lo tanto es una t-student de 18 grados de libertad $X_{\text{test}}|_{\mathbf{X}=\mathbf{x}} \sim t_{18}$ (véase Cuadro 1.2).

Tanto *a priori* como *a posteriori*, la variable T es una Gamma (la chi-cuadrado es un caso particular de Gamma). Este fenómeno de mantenerse dentro de una familia ocurre por cierta compatibilidad entre la distribución *a priori* y la verosimilitud (en este caso una normal). Cuando se da este fenómeno se dice que la distribución *a priori* es una **conjugada a priori**. Las soluciones analíticas suelen proponer conjugadas, como distribución

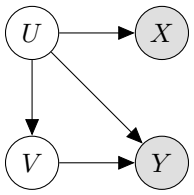


Figura 6.1: Ejemplo de red bayesiana, donde se puede apreciar tanto relaciones de causalidad como independencia.

a priori, ya que así garantizan que la distribución *a posteriori* pertenezca a una familia conocida (la misma que la distribución *a priori*). Es simplemente una recomendación para hacer sencillos (o al menos factibles) los cálculos. Los problemas de la estadística bayesiana, pueden representarse fácilmente con grafos denominados **redes bayesianas**.

6.1.1. Redes Bayesianas

Se denomina *modelo gráfico* a todo modelo probabilístico capaz de representarse con un grafo. En particular el modelado bayesiano es un modelo gráfico. Existen diferentes grafos que se pueden utilizar para describir los modelos, siendo las *redes bayesianas* el estándar en este tipo de estadística.

Se denomina red bayesiana a un grafo acíclico dirigido que representa la relación de causalidad e independencia de sus variables [7, Sección 3.5]. Por un lado, la causalidad está determinada por la dirección de sus vínculos y presenta una configuración a implementar para generar muestras del modelo (véase Sección 4.1). Por otro lado, dos variables aleatorias cualesquiera son condicionalmente independientes dados los valores de sus padres causales (y por lo tanto las raíces son independientes).

En la Fig. 6.1 puede verse un ejemplo de red bayesiana, donde el color gris hace referencia a las variables observables. La causalidad nos indica como podríamos simular el modelo:

- Muestrear U, ya que es raíz del grafo: $u \leftarrow U \sim p_U$.
- Muestrear $X|U = u$, ya que a su nodo le llega una conexión desde U: $x \leftarrow X \sim p_{X|U=u}$.
- Muestrear $V|U = u$: $v \leftarrow V \sim p_{V|U=u}$.
- Muestrear $Y|U = u, W = w$: $y \leftarrow Y \sim p_{Y|U=u,W=w}$.

Este análisis es conceptual, en la práctica no se habitual generar muestras de variables observables. Además, el procedimiento antes descripto nos define la factorización de la

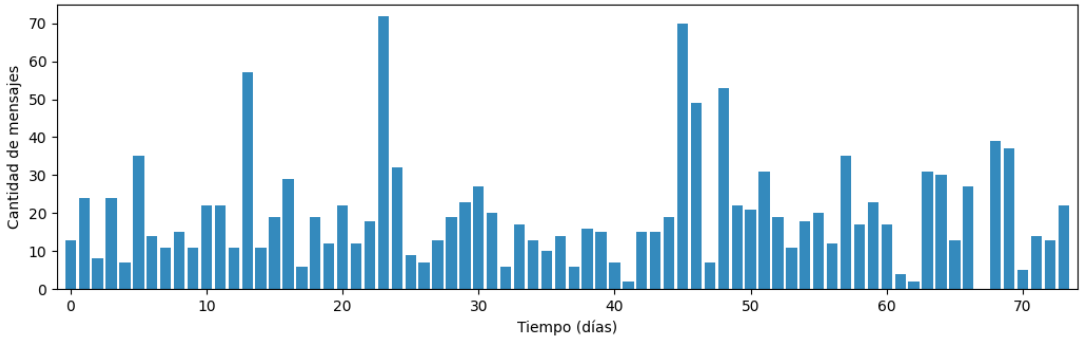


Figura 6.2: Ejemplo Sección 6.1.2, representa los mensajes recibidos durante los diferentes días. El ejemplo fue tomado de [6].

distribución conjunta:

$$p_{XYUV}(x, y, u, v) = p_U(u) \cdot p_{X|U=u}(x) \cdot p_{V|U=u}(v) \cdot p_{Y|U=u,W=w}(y) \tag{6.6}$$

Esta factorización nos impone condiciones de independencia. Por ejemplo, dado $U = u$, X y V son independientes ya que:

$$p_{XV|U=u}(x, v) = \frac{\int_y p_{XYUV}(x, y, u, v)dy}{p_U(u)} = p_{X|U=u}(x) \cdot p_{V|U=u}(v) \tag{6.7}$$

6.1.2. Ejemplo de Modelo Bayesiano

La esencia de la estadística bayesiana es interpreta la probabilidad como una medida de credibilidad en un evento, es dice buscar verdades en contexto de incertidumbre. La falta de certeza en las ciencias empíricas, lejos de volverlas absurdas, permite evitar afirmar más de lo que se sabe sin ocultar lo que efectivamente se conoce. Los métodos bayesianos no solo pueden adaptarse a intentar resolver los mismos problemas que la estadística frecuentista (por ejemplo predicciones), sino que también pueden intentar resolver problemas donde la estadística clásica es insuficiente o iluminar el sistema subyacente con un modelado más flexible. Veamos el siguiente ejemplo [6].

Un usuario proporciona una serie de recuentos diarios de mensajes de whatsapp enviados. Tiene curiosidad por saber si los hábitos de envío de mensajes han cambiado con el tiempo. En la Fig. 6.2 puede verse la cantidad de mensajes recibidos en los diferentes días. La hipótesis es que el arribo de mensajes tenía cierta tendencia y en algún momento cambio a otra diferente. Se desea plantear un modelo capaz de representar esta información.

La cantidad de mensajes en un día deberá ser modelada como una variable discreta cuyos átomos es $\mathbb{N}_0$. Por ejemplo, se elegirá una $X_i \sim \text{Poi}(\lambda_i)$. Esta será la única variable observable del modelo. Analizando los datos, parecería que el valor de λ_i aumenta en algún momento durante las observaciones. ¿Cómo podemos representar matemáticamente esta

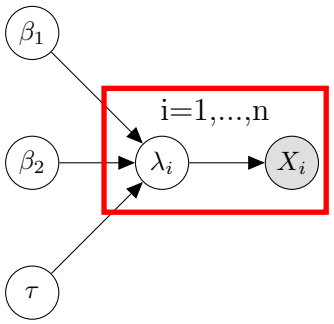


Figura 6.3: Red bayesiana del modelo propuesto para caracterizar el arribo de mensajes.

observación? Supongamos que algún día τ durante el período de observación, el parámetro λ_i se incrementa repentinamente. Entonces realmente tenemos dos tasas: una para el período anterior a τ y otro para el resto del período:

$$\lambda_i = \begin{cases} \beta_1 & i < \tau \\ \beta_2 & i \geq \tau \end{cases} \tag{6.8}$$

Tanto β_1 como β_2 toman valores reales no negativos. Por ejemplo, se puede elegir $\beta_1, \beta_2 \sim \mathcal{E}(\alpha)$. Es importante notar que se definen como variables aleatorias independientes e idénticamente distribuidas (para evitar sesgar a alguna tasa). En este punto, uno podría definir α como variable aleatoria o asignarle un valor fijo. Asignar un valor fijo para α sería menos influyente en el modelo que haberlo asignado en los λs, básicamente por estar más lejos de la variable observable a nivel grafo. Por el contrario, sería más influyente que suponerlo una variable aleatoria y asignarle un valor a los parámetros de esta nueva variable. Para este análisis, donde solamente se quiere estudiar el cambio de tasa, es suficiente con fijar un valor razonable. Notar que

$$\mathbb{E}[X_i] = \mathbb{E}\left[\mathbb{E}[X_i|\lambda_i]\right] = \begin{cases} \mathbb{E}\left[\beta_1\right] & i < \tau \\ \mathbb{E}\left[\beta_2\right] & i \geq \tau \end{cases} = \frac{1}{\alpha} \tag{6.9}$$

por lo que $\alpha = \frac{1}{\frac{1}{n}\sum_{i=1}^{n} X_i}$ es un buen candidato a valor. La última variable a definir es τ. Debido a la varianza de los datos, es difícil caracterizarla en detalle. Podemos asignar entonces la creencia menos informativa posibles *a priori* $\tau \sim \mathcal{U}\{1:n\}$, donde se asumirá τ independiente de β_1 y β_2.

En la Fig. 6.3 puede verse la red bayesiana del modelo descripto. La potencia del modelado bayesiano radica en poder modelar fácilmente situaciones prácticas donde la estadística clásica parece quedarse corta. El problema está en que calcular la distribución predictiva de un modelo de estas características es computacionalmente inviable, al menos

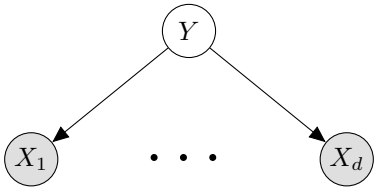


Figura 6.4: Red bayesiana del modelo gráfico bayes naive. Los predictores son independientes conocida la clase.

de forma exacta. En el resto del capítulo iremos analizando las fortalezas y debilidades de algunos modelos gráficos, llendo de las más simples a las más complejas incrementando el potencial poco a poco.

6.2. Bayes Naive

Uno de los modelos gráficos más fáciles de analizar son los conocido como **bayes naive**. La hipótesis naive en un problema de clasificación modela una relación de causalidad $Y \to \mathbf{X}$, donde las diferentes componentes $X_j|_{Y=k}$ son independientes, tal como puede verse en la red bayesiana de la Fig. 6.4. En este tipo de modelos, la probabilidad asignada a cada clase, cuando se observa una muestra, es de la forma:

$$p(y|\mathbf{x}) \propto p(y) \prod_{j=1}^{d} p(x_j|y) \tag{6.10}$$

Estos modelos se llaman **bayes naive** por combinar la hipótesis naive con la regla de bayes de (6.10). No es necesariamente un modelo bayesiano en el sentido de interpretar los parámetros como variables aleatorias ni mucho menos calcular una predictiva. De hecho, el cálculo de una predictiva en bayes naive se suele volverse prohibitivo ya que implica resolver la integral sobre una productoria [7, Sección 9.3].

Está claro que difícilmente en la práctica las componentes serán independientes dada la clase a la que pertenecen, de ahí radica el nombre de *naive* (ingenuo). Sin embargo, proponer modelos simples puede ser beneficioso incluso si no se cumplen el la práctica, ya que este tipo de modelos suele poseer menos parámetros y por lo tanto necesita menos datos para ser entrenado.

La simpleza de bayes naive, radica en la separación del modelo $X_j|_{Y=k}$ para cada clase. Nos permite modelar cada clase por separado para finalmente combinarlas por una $p(y)$ usualmente estimada con la proporción de muestras de entrenamiento de esa clase: $Y \sim \text{Cat}(\{c_1, \cdots, c_K\})$ con $c_k = \frac{\#\{y_i=k\}}{n}$. A continuación vamos a estudiar dos modelos bayes naive que utilizan estimadores puntuales: el primero de la estadística clásica y el segundo del modelado bayesiana propiamente dicha.

6.2.1. Bayes Naive Gaussiano

El modelo **bayes naive gaussiano** (GNB) propone modelar el problema como una mezcla de gaussianas naives: $Y \sim \text{Cat}(\{c_1, \cdots, c_K\})$ y $X_j|_{Y=k} \sim \mathcal{N}(\mu_j^{(k)}, \sigma_j^{2(k)})$, donde los parámetros serán estimados utilizando los estimadores estándar.

Este modelo está muy relacionado con los modelos de LDA y QDA estudiados anteriormente. Los tres métodos asumen un modelo de mezclas de gaussianas, su diferencia radica en las hipótesis que hacen sobre las matrices de covarianza, como puede verse en la Fig. 6.5. Sea Σ_k la matriz de covarianza asociadas a la clase k-ésima.

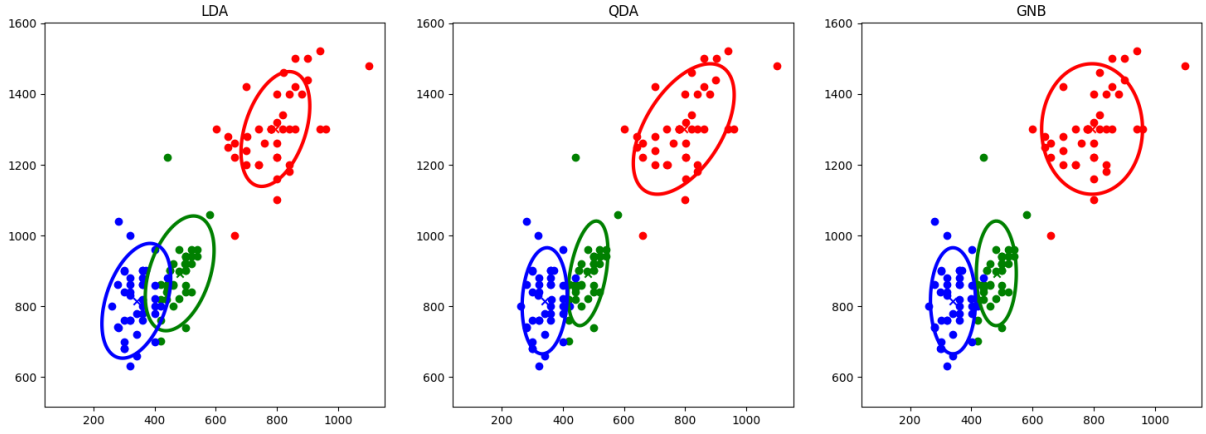
- QDA acepta como covarianza cualquier conjunto de matrices definidas positiva. Suelen estimarse como $\Sigma_k = \frac{1}{|\mathcal{D}_k|-1} \sum_{x \in \mathcal{D}_k} (x - \mu^{(k)}) \left(x - \mu^{(k)}\right)^T$.
- LDA acepta como coviarianza cualquier matriz (definida positiva) pero todas deben iguales. A partir de las matrices de covarianza de QDA, la covarianza de LDA puede computarse como $\Sigma = \frac{1}{n-K} \sum_{k=1}^{K} (|\mathcal{D}_k| - 1)\Sigma_k$.
- GNB permite tener matrices diferentes pero todas deben ser diagonales diagonales. A partir de las matrices de covarianza de QDA, las covarianzas de GNB pueden computarse como $\Sigma_k = \text{DIAG}\left(\sigma_1^{2(k)}, \cdots, \sigma_d^{2(k)}\right)$.

Por razones computacionales, en el caso de GNB, no se calcula toda la covarianza para luego quedarse con la raíz. En cambio, suele implementarse un cálculo de la forma $\sigma_j^{2(k)} = \frac{1}{|\mathcal{D}_k|-1} \sum_{x \in \mathcal{D}_k} (x_j - \mu_j^{(k)})^2$, pero el resultado final sería el mismo.

QDA es el modelo más completo de los tres, pero al necesitar estimar mayor cantidad de parámetros requiere mayor cantidad de muestras para ser entrenado correctamente. Entre GNB y LDA, la cantidad de parámetros será superior en un caso o en el otro según la relación entre la cantidad de clases K y la cantidad de predictores d. De cualquier manera, no solo es importante tener en cuenta la cantidad de parámetros, sino que tan bien representadas están las hiótesis en el conjunto de datos con el que se cuenta.

6.2.2. Bayes Naive Multinomial

El modelo **bayes naive multinomial** (MNB) propone modelar para cada clase $Y = k$ un proceso de Bernoulli generalizado $X_j|Y = k \sim \text{Cat}(\{\theta_1^{(k)}, \cdots, \theta_V^{(k)}\})$, donde se respeta la hiótesis naive. Hablamos de proceso en este caso, porque queremos definir un modelo capaz de entregar muestras de dimensión variable. Este tipo de modelado suele utilizarse en procesamiento de texto, donde cada texto posee una cantidad diferente de palabras. A su vez, supondremos que cada clase posee probabilidad c_k, es decir, $Y \sim \text{Cat}(\{c_1, \cdots, c_K\})$.



(a) (b) (c)

Figura 6.5: Comparación de algoritmos para un ejemplo de dos dimensiones. (a) Modelo LDA, (b) modelo QDA, y (c) modelo GNB. La diferencia de sus modelos radica en las covarianzas de cada clase.

Para una muestra d-dimensional $\mathbf{x} = (x_1, \cdots, x_d)$, (6.10) puede escribirse como:

$$p(y|\mathbf{x}) \propto c_y \cdot \prod_{j=1}^{d} \theta_{x_j}^{(y)} = c_y \cdot \prod_{m=1}^{V} \left(\theta_m^{(y)}\right)^{N_m} \tag{6.11}$$

donde N_m representa la cantidad de variables X_j que tomaron el valor m y son estadísticos suficientes en este modelo. El cambio de variable en la productoria nos permite iterar en los valores m que pueden tomar los predictores, en lugar necesitar iterar en la cantidad j de los mismos. Sea $\mathbf{N} = (N_1, \cdots, N_V)$, no solamente se puede notar que $\sum_{m=1}^{V} N_m = d$, sino que también nos define la distribución multinomial de estos estadísticos $\mathbf{N}|_{Y=k} \sim \mathcal{M}_n(d, [\theta_1^{(k)}, \cdots, \theta_V^{(k)}])$. El uso de este estadístico nos permite resolver la inferencia como una solución lineal de la log-probabilidad

$$\log p(y|\mathbf{x}) = \text{cte} + \log(c_y) + \sum_{m=1}^{V} N_m \log\left(\theta_m^{(y)}\right) \tag{6.12}$$

simplemente hay que haber estimado previamente los parámetros c_k y $(\theta_1^{(k)}, \cdots, \theta_V^{(k)})$ para todo $k = 1 \cdots, K$. Estos parámetros serán estimados en la etapa de entrenamiento.

6.2.2.1. Entrenamiento de MNB

La probabilidad de cada clase será simplemente estimada por la proporción de muestras que hay en el conjunto de entrenamiento $c_k = \frac{\#\{y_i=k\}}{n}$, como suele hacerse en cualquier modelo naive. Los θs en cambio serán modelados de forma bayesiana.

Supongamos que contamos con un datos $\{(\mathbf{N}_i, y_i)\}_{i=1}^{n}$, donde cada muestra $\mathbf{N}_i$ tendrá asociada una cantidad de predictores d diferente. Debido a la hipótesis naive, es válido separar los problemas por clase. Es decir que para cada clase k se utilizarán solamente

los datos con $\{y_i = k\}$ distribuidos según las probabilidades $(\theta_1^{(k)}, \cdots, \theta_V^{(k)})$. A su vez, dado que las variables N_m cuentan ocurrencias, puedo compactar todas las muestras de entrenamiento de cada clase en una sola, utilizando suficiencia estadística de las variables categóricas

$$\tilde{N}_m^{(k)} = \sum_{i=1}^{n} N_{i,m} \cdot \mathbf{1}\{y_i = k\} \tag{6.13}$$

Esto quiere decir por ejemplo, que en un problema de procesamiento de texto, podemos pensar que tenemos un solo texto por clase, concatenando todos los textos de dicha clase en uno solo. Esto implica que $(\tilde{N}_1^{(k)}, \cdots, \tilde{N}_V^{(k)})|_{\mathbf{T}_k=(\theta_1^{(k)},\cdots,\theta_V^{(k)})} \sim \mathcal{M}_n(\tilde{d}^{(k)}, [\theta_1^{(k)}, \cdots, \theta_V^{(k)}])$, donde $\tilde{d}^{(k)}$ representa la cantidad de "palabras" que posee el "texto concatenado" de la clase k. Notar que hemos definido una variable aleatoria $\mathbf{T}_k$ representativa de los parámetros del modelo. Dicha variable será modelada *a priori* como $\mathbf{T}_k \sim \text{Dir}([\alpha_1, \cdots, \alpha_V])$, donde el vector aleatorio tendrá distribución Dirichlett.

Definición 6.1 *El vector aleatorio $(\theta_1, \cdots, \theta_V)$ tiene distribución Dirichlett de parámetros $(\alpha_1, \cdots, \alpha_V)$ si su densidad conjunta es de la forma:*

$$p(\theta_1, \cdots, \theta_V) = \frac{\prod_{m=1}^{V} \Gamma(\alpha_m)}{\Gamma\left(\sum_{m=1}^{V} \alpha_m\right)} \left(\prod_{m=1}^{V} \theta_m^{\alpha_m - 1}\right) \cdot \mathbf{1}\left\{\sum_{m=1}^{V} \theta_m = 1, \theta_m \geq 0\right\} \tag{6.14}$$

La principal propiedad de este tipo de vectores es que sus marginales posee **distribución beta** $T_m \sim \beta(\alpha_m, \sum_{\eta \neq m} \alpha_\eta)$, una variable aleatoria utilizada para modelar probabilidades cuya esperanza es $\mathbb{E}[T_m] = \frac{\alpha_m}{\sum_{\eta=1}^{V} \alpha_\eta}$. Con este modelo, la *distribución a posteriori* puede calcularse como:

$$\begin{aligned}
&p\left(\theta_1^{(k)}, \cdots, \theta_V^{(k)} | \tilde{N}_1^{(k)}, \cdots, \tilde{N}_V^{(k)}\right) \\
&\quad \propto P\left(\tilde{N}_1^{(k)}, \cdots, \tilde{N}_V^{(k)} | \theta_1^{(k)}, \cdots, \theta_V^{(k)}\right) \cdot p\left(\theta_1^{(k)}, \cdots, \theta_V^{(k)}\right) && (6.15) \\
&\quad \propto \left(\prod_{m=1}^{V} (\theta_m^{(k)})^{\tilde{N}_m^{(k)}}\right) \left(\prod_{m=1}^{V} (\theta_m^{(k)})^{\alpha_m - 1} \cdot \mathbf{1}\left\{\theta_m^{(k)} \geq 0\right\}\right) \cdot \mathbf{1}\left\{\sum_{m=1}^{V} \theta_m^{(k)} = 1\right\} && (6.16)
\end{aligned}$$

con lo cuál $\mathbf{T}_k|_{\tilde{N}_1^{(k)},\cdots,\tilde{N}_V^{(k)}} \sim \text{Dir}([\tilde{N}_1^{(k)} + \alpha_1, \cdots, \tilde{N}_V^{(k)} + \alpha_V])$. De esta manera, utilizando como estimador puntual bayesiano la **media a posteriori** se obtiene que

$$\theta_m^{(k)} = \mathbb{E}[T_m | \tilde{N}_1^{(k)}, \cdots, \tilde{N}_V^{(k)}] = \frac{\tilde{N}_m^{(k)} + \alpha_m}{\sum_{\eta=1}^{V} \tilde{N}_\eta^{(k)} + \alpha_\eta}. \tag{6.17}$$

Mientras que GNB es un modelo gráfico con estimadores puntuales clásicos, MNB utiliza estimadores puntuales bayesianos. Sin embargo, todavía nos estamos conformando con hacer una estimación puntual en lugar de un cálculo predictivo. A continuación veremos cuanto es necesario complejizar al modelo para efectivamente poder hacer dicho cálculo.

6.3. Bayes Variacional Gaussiano

La estadística bayesiana basa su supuesto en considerar a los parámetros parte del espacio latente, los que se sumarán a las variables no observables propias del modelo (como la variable mezcladora en un problema de *clustering*). Sea $\mathbf{Z}$ el vector de variables no observable, en general será prohibitivo calcular la distribución *a posteriori* $p(\mathbf{z}|\mathbf{x})$ en modelos complejos. Una alternativa es aproximar dicha distribución minimizando la **divergencia de Kullback Leibler**:

$$\arg\min_{q\in\mathcal{P}} \ \mathrm{KL}\left(q(\cdot|\mathbf{x})\|p(\cdot|\mathbf{x})\right) \tag{6.18}$$

donde $q(\mathbf{z}|\mathbf{x})$[1] cumple ciertas restricciones $\mathcal{P}$ que permitan limitar el modelo posible de forma que sea factible emplear un estudio analítico. Dicha divergencia puede descomponerse como $\mathrm{KL}\left(q(\cdot|\mathbf{x})\|p(\cdot|\mathbf{x})\right) = \log p(\mathbf{x}) - \mathrm{ELBO}(q(\cdot|\mathbf{x}))$, donde

$$\mathrm{ELBO}(q(\cdot|\mathbf{x})) = H(q(\cdot|\mathbf{x})) + \mathbb{E}_q\left[\log p(\mathbf{x},\mathbf{Z})|\mathbf{X}=\mathbf{x}\right] \tag{6.19}$$

donde $H(q(\cdot|\mathbf{x}))$ representa la entropía como función de $\mathbf{x}$. Similar al algoritmo EM, la **cota inferior esperada** (ELBO) acota la verosimilitud de forma maximizar una cota fomente la maximización de la misma $\mathrm{ELBO}(q(\cdot|\mathbf{x})) \leq \log p(\mathbf{x})$ (ya que la divergencia de Kullback Leibler es no negativa). Esta log-verosimilitud debe ser entendida por muestra, por lo que la para la log-verosimilitud de un conjunto simplemente se sumarían los ELBOs. A su vez, este mismo fenómeno se da acotando a:

$$\mathrm{ELBO}(q) = \mathbb{E}_q\left[-\log p(\mathbf{x}|\mathbf{z})|\mathbf{x}\right] - \mathrm{KL}\left(q(\cdot|\mathbf{x})\|p(\cdot)\right) \leq \mathbb{E}_q\left[-\log p(\mathbf{x}|\mathbf{z})|\mathbf{x}\right] \tag{6.20}$$

donde $\mathbb{E}_q\left[-\log p(\mathbf{x}|\mathbf{z})|\mathbf{x}\right]$ es la *entropía cruzada* asociada a un **autoencoder** de *encoder* $q(\cdot|\mathbf{x})$ y *decoder* $p(\cdot|\mathbf{z})$. Es decir que la maximización de la ELBO también tenderá a aumentar dicha magnitud.

Minimizar la divergencia de Kullback Leibler (6.18) equivale a maximizar la ELBO. Para poder tratar el problema, se suele asumir como restricción sobre $\mathcal{P}$ la llamada **Mean field approximation**: Suponer que q se puede factorizar como productos de densidades tratables, separando entre las variables ocultas $\mathbf{u}$ y los parámetros ϕ. Es decir, sea $\mathbf{z} = (\mathbf{u},\phi)$ se relaja el problema suponiendo $q(\mathbf{z}|\mathbf{x}) = q_1(\mathbf{u}|\mathbf{x})q_2(\phi|\mathbf{x})$ para todo $q \in \mathcal{P}$. Con esta hipótesis, la entropía de la distribución producto se descompone como suma $H(q(\cdot|\mathbf{x})) = H(q_1(\cdot|\mathbf{x})) + H(q_2(\cdot|\mathbf{x}))$ y la esperanza del logaritmo de la conjunta se puede descomponer en dos sentidos:

$$\mathbb{E}_q\left[\log p(\mathbf{x},\mathbf{Z})|\mathbf{X}=\mathbf{x}\right] = \int_{\mathcal{U}} q_1(\mathbf{u}|\mathbf{x}) \left(\int_{\Phi} q_2(\phi|\mathbf{x}) \log p(\mathbf{x},\mathbf{u},\phi) d\phi\right) d\mathbf{u} \tag{6.21}$$

$$= \int_{\Phi} q_2(\phi|\mathbf{x}) \left(\int_{\mathcal{U}} q_1(\mathbf{u}|\mathbf{x}) \log p(\mathbf{x},\mathbf{u},\phi) d\mathbf{u}\right) d\phi \tag{6.22}$$

El algoritmo **bayes variacional** propone resolver el problema (6.18) de forma iterativa:

[1] Notar que estamos haciendo un análisis para toda la muestra $\mathbf{x}$.

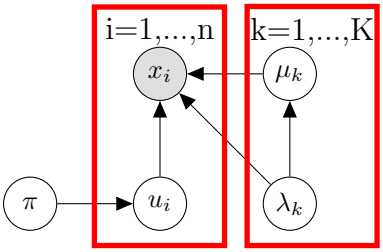


Figura 6.6: Red bayesiana del modelo bayes variacional gaussiano.

suponer q_1 fijo para optimizar en q_2 para luego dejar fijo q_2 para optimizar en q_1.

$$q_1(\cdot|\mathbf{x}) = \arg\max_{q_1} \int_{\mathcal{U}} q_1(\mathbf{u}|\mathbf{x}) \log \frac{e^{E_1(\mathbf{x},\mathbf{u})}}{q_1(\mathbf{u}|\mathbf{x})} d\mathbf{u} \tag{6.23}$$

$$q_2(\cdot|\mathbf{x}) = \arg\max_{q_2} \int_{\Phi} q_2(\mathbf{u}|\mathbf{x}) \log \frac{e^{E_2(\mathbf{x},\phi)}}{q_2(\phi|\mathbf{x})} d\phi \tag{6.24}$$

donde

$$E_1(\mathbf{x},\mathbf{u}) = \int_{\Phi} q_2(\phi|\mathbf{x}) \log p(\mathbf{x},\mathbf{u},\phi) d\phi \equiv f(q_2) \tag{6.25}$$

$$E_2(\mathbf{x},\phi) = \int_{\mathcal{U}} q_1(\mathbf{u}|\mathbf{x}) \log p(\mathbf{x},\mathbf{u},\phi) d\mathbf{u} \equiv f(q_1) \tag{6.26}$$

El problema de optimización (6.23) equivale a minimizar la divergencia de Kullback Leibler entre $q_1(\mathbf{u}|\mathbf{x})$ y $k_1 e^{E_1(\mathbf{x},\mathbf{u})}$, donde k_1 es una constante de proporcionalidad para que la expresión sea una densidad/probabilidad en $\mathbf{u}$. De manera similar ocurre con $q_2(\phi|\mathbf{x})$ y $k_2 e^{E_2(\mathbf{x},\phi)}$ para el problema (6.24). Entonces, el algoritmo bayes varaicional consiste en iterar entre $q_1(\mathbf{u}|\mathbf{x}) \propto e^{E_1(\mathbf{x},\mathbf{u})}$ y $q_2(\phi|\mathbf{x}) \propto e^{E_2(\mathbf{x},\phi)}$.

6.3.1. Mezcla de Gaussianas escalares en Bayes Variacional

Para entender las complicaciones para efectuar los cálculos, incluso en un problema simple, supongamos el modelo de mezcla de gaussianas escalares graficado en la Fig. 6.6. Este problema se lo conoce como **Bayes Variacional Gaussiano** (GVB) [20, Sección 10.2]. Interpretando su red bayesiana, la distribución del modelo puede escribirse como

$$p(\mathbf{x},\mathbf{u},\pi,\lambda,\mu) = p(\pi)\left(\prod_{k=1}^{K} p(\lambda_k)p(\mu_k|\lambda_k)\right)\left(\prod_{i=1}^{n} P(u_i|\pi)p(x_i|u_i,\mu,\lambda)\right) \tag{6.27}$$

donde $u|\pi \sim \text{Cat}(\pi)$ y $x|u,\mu,\lambda \sim \mathcal{N}(\mu_u, \lambda_u^{-1})$. *A priori* supondremos distribuciones conjugadas a priori con la verosimilitud: $\pi \sim \text{Dir}(\alpha)$, $\lambda_k \sim \Gamma(\nu,\beta)$ y $\mu_k|\lambda_k \sim \mathcal{N}(m,(\delta\lambda_k)^{-1})$. Los parámetros de este modelo serán entonces α,ν,β,m y δ. En este caso $\mathbf{u} = (u_1,\cdots,u_n)$ y $\phi = (\pi,\mu,\lambda)$ con $\pi = (\pi_1,\cdots,\pi_K)$, $\mu = (\mu_1,\cdots,\mu_K)$ y $\lambda = (\lambda_1,\cdots,\lambda_K)$. Un caso particular cuando $K = 1$ y $\delta \to \infty$ fue estudiando en el Ej. 6.1, donde la distribución a posteriori de la precisión (inversa de la varianza) es una Gamma y la distribución predic-

tiva es una t-student.

Utilizar como distribución *a priori* una media gaussiana y una precisión Gamma, es una familia muy estudiada en la bibliografía y recibe el nombre de **distribución normal-gamma**.

Definición 6.2 *Un vector aleatorio normal-gamma* (ν, β, m, δ) *tiene una densidad conjunta de la forma:*

$$p(\lambda, \mu) = \frac{\beta^\nu}{\Gamma(\nu)} \sqrt{\frac{\delta}{2\pi}} \lambda^{\nu - \frac{1}{2}} e^{-\lambda\left(\beta + \frac{\delta\mu^2}{2} - \delta m\mu + \frac{\delta m^2}{2}\right)} \mathbf{1}\{\lambda > 0\} \tag{6.28}$$

donde $\lambda \sim \Gamma(\nu, \beta)$ *y* $\mu|\lambda \sim \mathcal{N}(m, (\delta\lambda)^{-1})$.

Teniendo en cuenta todas estas definiciones, a continaución se procederá a calcular la distribución *a posteriori* y la predictiva del modelo.

6.3.1.1. Distribución a posteriori en GVB

Bajo la hipótesis *mean field approximation*, $q(\mathbf{u}, \pi, \lambda, \mu|\mathbf{x}) = Q_1(\mathbf{u}|\mathbf{x}) q_2(\pi, \lambda, \mu|\mathbf{x})$, por lo que

$$E_1(\mathbf{x}, \mathbf{u}) = \text{cte} + \sum_{i=1}^{n} \int q_2(\pi|\mathbf{x}) \log P(u_i|\pi) d\pi + \sum_{i=1}^{n} \int\int q_2(\mu, \lambda|\mathbf{x}) \log p(x_i|u_i, \mu, \lambda) d\mu d\lambda \tag{6.29}$$

donde "cte" hace referencia términos que no dependen de q_2. De forma análoga

$$\begin{aligned} E_2(\mathbf{x}, \pi, \lambda, \mu) = &\log p(\pi) + \sum_{k=1}^{K} \log p(\lambda_k) + \sum_{k=1}^{K} \log p(\mu_k|\lambda_k) \\ &+ \sum_{i=1}^{n} \sum_{k=1}^{K} Q_1(u_i = k|\mathbf{x}) \log P(u_i = k|\pi) \\ &+ \sum_{i=1}^{n} \sum_{k=1}^{K} Q_1(u_i = k|\mathbf{x}) \log p(x_i|u_i = k, \mu, \lambda) \end{aligned} \tag{6.30}$$

En primer lugar, se considerará Q_1 fijo y conocido para computar q_2. Esto se conoce como **actualización de los parámetros a posteriori**. Es decir, asumiendo que las familias elegidas son conjuntadas a priori, los parámetros *a priori* son α, ν, β, m y δ tendrán sus contra-partes *a posteriori* $\alpha^*, \nu^*, \beta^*, m^*$ y δ^* (los cuales puede ser diferentes para cada clase). Lo primero a notar es la factorización. Sea $\gamma_{i,k} = Q_1(u_i = k|\mathbf{x})$, luego $q_2(\pi, \lambda, \mu|\mathbf{x}) \propto e^{E_2(\mathbf{x}, \pi, \lambda, \mu)}$:

$$q_2(\pi, \lambda, \mu|\mathbf{x}) \propto p(\pi) \left(\prod_{k=1}^{K} p(\lambda_k) p(\mu_k|\lambda_k) \right) \prod_{k=1}^{K} e^{\sum_{i=1}^{n} \gamma_{i,k} [\log \pi_k + \log \mathcal{N}(x_i|\mu_k, \lambda_k^{-1})]} \tag{6.31}$$

y por lo tanto $q_2(\pi, \lambda, \mu|\mathbf{x}) = q_2(\pi|\mathbf{x}) \prod_{k=1}^{K} q_2(\mu_k, \lambda_k|\mathbf{x})$ (es decir, se acaba de demostrar la relación de independiencia, resta calcular $q_2(\pi|\mathbf{x})$ y $q_2(\mu_k, \lambda_k|\mathbf{x})$ para cada k). Dado que

el logaritmo de la densidad normal es una magnitud cuadrática en las muestras, podemos definir los siguientes **estadísticos suficientes**:

$$N_k = \sum_{i=1}^{n} \gamma_{i,k}, \qquad f_k = \sum_{i=1}^{n} \gamma_{i,k} x_i, \qquad s_k = \sum_{i=1}^{n} \gamma_{i,k} x_i^2 \tag{6.32}$$

y por lo tanto, el último exponente de (6.31) puede simplificarse como:

$$N_k \log \pi_k - \frac{N_k}{2} \log(2\pi) + \frac{N_k}{2} \log(\lambda_k) - \frac{\lambda_k}{2}(s_k - 2 f_k \mu_k + N_k \mu_k^2) \tag{6.33}$$

Para computar $q_2(\pi|\mathbf{x})$, basta por juntar los términos de (6.31) donde aparece π

$$q_2(\pi|\mathbf{x}) \propto \left(\prod_{k=1}^{K} \pi_k^{\alpha_k - 1} e^{N_k \log \pi_k} \right) \mathbf{1}\left\{ \sum_{k=1}^{K} \pi_k = 1, \pi_k \geq 0 \right\} \tag{6.34}$$

con lo cual $\pi|\mathbf{x} \sim \text{Dir}([\alpha_1 + N_1, \cdots, \alpha_K + N_K])$, es decir $\alpha_k^* = \alpha_k + N_k$. Para el cálculo de $q_2(\mu_k, \lambda_k|\mathbf{x})$, resta analizar:

$$q_2(\mu_k, \lambda_k|\mathbf{x}) \propto \underbrace{\lambda_k^{\nu-1} e^{-\beta\lambda_k} \mathbf{1}\{\lambda_k > 0\}}_{\propto p(\lambda_k)} \underbrace{\lambda_k^{1/2} e^{\frac{-\delta\lambda_k(\mu_k - m)^2}{2}}}_{\propto p(\mu_k|\lambda_k)} \lambda_k^{\frac{N_k}{2}} e^{\frac{-\lambda_k(s_k - 2 f_k \mu_k + N_k \mu_k^2)}{2}} \tag{6.35}$$

$$\propto \lambda_k^{\nu + \frac{N_k}{2} - \frac{1}{2}} e^{-\lambda_k \left(\beta + \frac{\delta\mu_k^2}{2} - \delta m \mu_k + \frac{\delta m^2}{2} + \frac{s_k}{2} - f_k \mu_k + \frac{N_k \mu_k^2}{2} \right)} \mathbf{1}\{\lambda_k > 0\} \tag{6.36}$$

Se puede apreciar que esta distribución es una normal-gamma. Para calcular los parámetros *a posteriori* basta con comparar la expresión con (6.28) (con los parámetros a posteriori $\nu_k^*, \beta_k^*, m_k^*, \delta_k^*$) y despejar:

- $\nu_k^* - \frac{1}{2} = \nu + \frac{N_k}{2} - \frac{1}{2}$.
- $\frac{\delta_k^*}{2} = \frac{\delta + N_k}{2}$.
- $\delta_k^* m_k^* = \delta m + f_k$.
- $\beta_k^* + \frac{\delta_k^* m_k^{*2}}{2} = \beta + \frac{\delta m^2}{2} + \frac{s_k}{2}$.

Es decir, $\nu_k^* = \nu + \frac{N_k}{2}$, $\delta_k^* = \delta + N_k$ y $m_k^* = \frac{\delta m + f_k}{\delta + N_k}$. Para el caso de β_k^*, se puede calcular como:

$$\beta_k^* = \beta + \frac{\delta m^2}{2} + \frac{s_k}{2} - \frac{(\delta m + f_k)^2}{2(\delta + N_k)} \tag{6.37}$$

con lo cual $\mu_k|\lambda_k, \mathbf{x} \sim \mathcal{N}\left(\frac{\delta m + f_k}{\delta + N_k}, \frac{1}{\lambda_k(\delta + N_k)} \right)$ y $\lambda_k|\mathbf{x} \sim \Gamma\left(\nu + \frac{1}{2} N_k, \beta + \frac{\delta m^2}{2} + \frac{1}{2} s_k - \frac{(\delta m + f_k)^2}{2(\delta + N_k)} \right)$.

En segundo lugar, se considerará q_2 fijo y conocido para calcular Q_1. Esto se conoce como **actualización de la distribución de las variables ocultas**. Para calcular dicha distribución, se utilizarán las siguientes propiedades de las distribuciones Gamma y Beta.

Propiedades 6.1 *Sean $\lambda \sim \Gamma(\nu, \beta)$ y $\pi \sim \beta(a, b)$, la esperanza del logaritmo de estas variables se calcula como $\mathbb{E}[\log \lambda] = \psi(\nu) - \log(\beta)$ y $\mathbb{E}[\log \pi] = \psi(a) - \psi(a + b)$,*

donde $\psi(\cdot)$ es la función digamma $\psi(z) = \frac{\Gamma'(z)}{\Gamma(z)}$.

Lo primero a notar para el cómputo de $Q_1(\mathbf{u}|\mathbf{x}) \propto e^{E_1(\mathbf{x},\mathbf{u})}$ es la independencia de sus variables:

$$Q_1(\mathbf{u}|\mathbf{x}) \propto \prod_{i=1}^{n} e^{\int q_2(\pi|\mathbf{x}) \log P(u_i|\pi) d\pi + \int\int q_2(\mu,\lambda|\mathbf{x}) \log p(x_i|u_i,\mu,\lambda) d\mu d\lambda} = \prod_{i=1}^{n} Q_1(u_i|\mathbf{x}) \tag{6.38}$$

entonces basta con calcular cada $Q_1(u_i = k|\mathbf{x})$. Sean los parámetros de q_2 definidos como $\pi|\mathbf{x} \sim \text{Dir}(\alpha^*)$, $\mu_k|\lambda_k, \mathbf{x} \sim \mathcal{N}\left(m_k^*, (\delta_k^*\lambda_k)^{-1}\right)$ y $\lambda_k|\mathbf{x} \sim \Gamma\left(\nu_k^*, \beta_k^*\right)$. Luego

$$Q_1(u_i = |\mathbf{x}) \propto e^{\psi(\alpha_k^*) - \psi\left(\sum_{c=1}^{K} \alpha_c^*\right) + \frac{\psi(\nu_k^*) - \log(\beta_k^*)}{2} - \frac{1}{2}\mathbb{E}_{q_2}[\lambda_k (x_i - \mu_k)^2 | \mathbf{x}]} \tag{6.39}$$

donde se usó que $\pi_k|\mathbf{x} \sim \beta(\alpha_k^*, \sum_{\eta \neq k} \alpha_\eta^*)$. La esperanza de la última expresión puede calcularse con

$$\mathbb{E}_{q_2}[\lambda_k (x_i - \mu_k)^2 | \mathbf{x}] = \mathbb{E}_{q_2}[\lambda_k \mathbb{E}_{q_2}[(x_i - \mu_k)^2 | \lambda_k, \mathbf{x}] | \mathbf{x}] \tag{6.40}$$

$$= \mathbb{E}_{q_2}[\lambda_k \mathbb{E}_{q_2}[(x_i - m_k^* + m_k^* - \mu_k)^2 | \lambda_k, \mathbf{x}] | \mathbf{x}] \tag{6.41}$$

$$= \mathbb{E}_{q_2}[\lambda_k \left((x_i - m_k^*)^2 + \mathbb{E}_{q_2}[(\mu_k - m_k^*)^2 | \lambda_k, \mathbf{x}] + 2(x_i - m_k^*)\mathbb{E}_{q_2}[m_k^* - \mu_k | \mathbf{x}]\right) | \mathbf{x}] \tag{6.42}$$

$$= \frac{\nu_k^*}{\beta_k^*}(x_i - m_k^*)^2 + \frac{1}{\delta_k^*} + 0 \tag{6.43}$$

Finalmente, se puede despejar una fórmula para las probabilidades:

$$Q_1(u_i = k|\mathbf{x}) \propto e^{\psi(\alpha_k^*) - \psi\left(\sum_{c=1}^{K} \alpha_c^*\right) + \frac{\psi(\nu_k^*) - \log(\beta_k^*)}{2} - \frac{1}{2\delta_k^*} - \frac{\nu_k^*}{2\beta_k^*}(m_k^* - x_i)^2} \tag{6.44}$$

En resumen, la distribución *a posteriori* en un problema de GVB se calcula inicializando $\gamma_{i,k}$ (por ejemplo con el algoritmo EM) e iterando entre:

- Calcular $(\alpha_k^*, m_k^*, \delta_k^*, \nu_k^*, \beta_k^*)$ a partir de $\gamma_{i,k}$ como

$$\alpha_k^* = \alpha_k + \sum_{i=1}^{n} \gamma_{i,k}, \quad m_k^* = \frac{\delta m + \sum_{i=1}^{n} \gamma_{i,k} x_i}{\delta + \sum_{i=1}^{n} \gamma_{i,k}} \tag{6.45}$$

$$\delta_k^* = \delta + \sum_{i=1}^{n} \gamma_{i,k}, \quad \nu_k^* = \nu + \frac{1}{2}\sum_{i=1}^{n} \gamma_{i,k} \tag{6.46}$$

$$\beta_k^* = \beta + \frac{\delta m^2}{2} + \frac{1}{2}\sum_{i=1}^{n} \gamma_{i,k} x_i^2 - \frac{(\delta m + \sum_{i=1}^{n} \gamma_{i,k} x_i)^2}{2(\delta + \sum_{i=1}^{n} \gamma_{i,k})} \tag{6.47}$$

- Calcular $\gamma_{i,k} = \frac{\rho_{i,k}}{\sum_{c=1}^{K} \rho_{i,c}}$ a partir de $(\alpha^*, m_k^*, \delta_k^*, \nu_k^*, \beta_k^*)$ como

$$\rho_{i,k} = e^{\psi(\alpha_k^*) - \psi\left(\sum_{c=1}^{K} \alpha_c^*\right) + \frac{\psi(\nu_k^*) - \log(\beta_k^*)}{2} - \frac{1}{2\delta_k^*} - \frac{\nu_k^*}{2\beta_k^*}(m_k^* - x_i)^2} \tag{6.48}$$

6.3.1.2. Distribución Predictiva en GVB

Por un lado, la distribución *a posteriori* q_2 estimada durante el entrenamiento cumple la relación de independencia $q_2(\pi, \lambda, \mu|\mathbf{x}) = q_2(\pi|\mathbf{x}) \prod_{k=1}^{K} q_2(\mu_k, \lambda_k|\mathbf{x})$. Por el otro, una nueva muestra x_{test} (que no pertenezca al conjunto de entrenaimento $\mathbf{x}$) pertenecerá a un modelo de mezcla $p(x_{\text{test}}|\pi, \mu, \lambda) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x_{\text{test}}|\mu_k, \lambda_k^{-1})$. Juntando ambas consideraciones

podemos ver que la distribución predictiva tambbién es una mezcla:

$$p(x_{\text{test}}|\mathbf{x}) = \mathbb{E}[p(x_{\text{test}}|\phi)|\mathbf{x}] \tag{6.49}$$

$$= \sum_{k=1}^{K} \mathbb{E}[\pi_k|\mathbf{x}] \cdot \mathbb{E}\left[\sqrt{\frac{\lambda_k}{2\pi}} e^{\frac{-\lambda_k}{2}(x_{\text{test}}-\mu_k)^2} \middle| \mathbf{x} \right] \tag{6.50}$$

$$= \sum_{k=1}^{K} \frac{\alpha_k^*}{\sum_{c=1}^{K} \alpha_c^*} \cdot \tilde{p}_k(x_{\text{test}}|\mathbf{x}) \tag{6.51}$$

donde se utilizó que $\pi_k|\mathbf{x} \sim \beta(\alpha_k^*, \sum_{\eta \neq k} \alpha_\eta^*)$. La distribución predictiva es una mezcla cuyos pesos son la proporción de los α^*; resta calcular las nuevas densidades $\tilde{p}_k(x_{\text{test}}|\mathbf{x})$. Para ello, notar que $(\mu_k, \lambda_k)|\mathbf{x}$ tienen una distribución normal-gamma (6.28) de parámetros $(\nu_k^*, \beta_k^*, m_k^*, \delta_k^*)$:

$$\tilde{p}_k(x_{\text{test}}|\mathbf{x}) \propto \int_0^\infty \int_{-\infty}^\infty \sqrt{\lambda_k} e^{\frac{-\lambda_k}{2}(x_{\text{test}}-\mu_k)^2} \lambda_k^{\nu_k^*-\frac{1}{2}} e^{-\lambda_k\left(\beta_k^* + \frac{\delta_k^*\mu_k^2}{2} - \delta_k^* m_k^* \mu_k + \frac{\delta_k^* {m_k^*}^2}{2}\right)} d\mu_k d\lambda_k \tag{6.52}$$

$$\propto \int_0^\infty \int_{-\infty}^\infty \lambda^{\nu_k^*} e^{-\lambda_k\left(\beta_k^* + \frac{\delta_k^*\mu_k^2}{2} - \delta_k^* m_k^* \mu_k + \frac{\delta_k^* {m_k^*}^2}{2} + \frac{\mu_k^2}{2} - x_{\text{test}}\mu_k + \frac{x_{\text{test}}^2}{2}\right)} d\mu_k d\lambda_k \tag{6.53}$$

Nuevamente el interior de la integral es una distribución normal-gamma. Como toda densidad integra 1, basta con reconocer los parámetros $(\tilde{\nu}, \tilde{\beta}, \tilde{m}, \tilde{\delta})$ de la nueva distribución para resolver la integral. Igualando con (6.28), se obtiene:

- $\tilde{\nu} - \frac{1}{2} = \nu_k^*$
- $\frac{\tilde{\delta}}{2} = \frac{\delta_k^*}{2} + \frac{1}{2}$
- $\tilde{\delta}\tilde{m} = \delta_k^* m_k^* + x_{\text{test}}$
- $\tilde{\beta} + \frac{\tilde{\delta}\tilde{m}^2}{2} = \beta_k^* + \frac{\delta_k^* {m_k^*}^2}{2} + \frac{x_{\text{test}}^2}{2}$

De las primeras tres ecuaciones es inmediato notar que $\tilde{\nu} = \nu_k^* + \frac{1}{2}$, $\tilde{\delta} = \delta_k^* + 1$ y $\tilde{m} = \frac{x_{\text{test}} + \delta_k^* m_k^*}{\delta_k^* + 1}$. Para la ecuación de $\tilde{\beta}$ basta con notar que

$$\tilde{\beta} = \beta_k^* + \frac{\delta_k^* {m_k^*}^2}{2} + \frac{x_{\text{test}}^2}{2} - \frac{(x_{\text{test}} + \delta_k^* m_k^*)^2}{2(\delta_k^* + 1)} \tag{6.54}$$

$$= \beta_k^* + \frac{(\delta_k^* {m_k^*}^2 + x_{\text{test}}^2)(\delta_k^* + 1) - (x_{\text{test}} + \delta_k^* m_k^*)^2}{2(\delta_k^* + 1)} \tag{6.55}$$

$$= \beta_k^* + \frac{{\delta_k^*}^2 {m_k^*}^2 + \delta_k^* {m_k^*}^2 + \delta_k^* x_{\text{test}}^2 + x_{\text{test}}^2 - x_{\text{test}}^2 - 2\delta_k^* m_k^* x_{\text{test}} - {\delta_k^*}^2 {m_k^*}^2}{2(\delta_k^* + 1)} \tag{6.56}$$

$$= \beta_k^* + \frac{\delta_k^* (x_{\text{test}} - m_k^*)^2}{2(\delta_k^* + 1)} \tag{6.57}$$

Como en la única variable que aparece x_{test} es en $\tilde{\beta}$, podemos decir que la integral es $\tilde{p}_k(x_{\text{test}}|\mathbf{x}) \propto \tilde{\beta}^{-\tilde{\nu}}$, es decir:

$$\tilde{p}_k(x_{\text{test}}|\mathbf{x}) \propto \left(\beta_k^* + \frac{\delta_k^* (x_{\text{test}} - m_k^*)^2}{2(\delta_k^* + 1)} \right)^{-(\nu_k^* + 1/2)} \tag{6.58}$$

$$\propto \left(1 + \frac{\delta_k^* \nu_k^*}{(\delta_k^* + 1)\beta_k^*} \frac{(x_{\text{test}} - m_k^*)^2}{2\nu_k^*}\right)^{-\frac{2\nu_k^* + 1}{2}} \tag{6.59}$$

Este tipo distribución se conoce como t-student generalizada.

Definición 6.3 *Una variable aleatoria tiene distribución t-Student Generalizada:* $X \sim t(\mu, \Lambda, \nu)$ *si su densidad es de la forma*

$$p(x) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \sqrt{\frac{\Lambda}{\pi\nu}} \left(1 + \Lambda \frac{(x-\mu)^2}{\nu}\right)^{-\frac{\nu+1}{2}} \tag{6.60}$$

Por lo tanto, la densidad predictiva es una mezcla de t-students de la forma:

$$p(x_{\text{test}}|\mathbf{x}) = \sum_{k=1}^{K} \frac{\alpha_k^*}{\sum_{c=1}^{K} \alpha_c^*} \cdot t\left(x_{\text{test}} \middle| m_k^*, \frac{\delta_k^* \nu_k^*}{(\delta_k^* + 1)\beta_k^*}, 2\nu_k^*\right) \tag{6.61}$$

Al ser una mezcla, esta distribución permite hacer predicciones (en el sentido frecuentista de la palabra) para el problema de clustering asociado $\tilde{p}(k|x_{\text{test}}, \mathbf{x})$ donde

$$\tilde{p}(k|x_{\text{test}}, \mathbf{x}) \propto \frac{\alpha_k^*}{\sum_{c=1}^{K} \alpha_c^*} \cdot t\left(x_{\text{test}} \middle| m_k^*, \frac{\delta_k^* \nu_k^*}{(\delta_k^* + 1)\beta_k^*}, 2\nu_k^*\right) \tag{6.62}$$

donde en el caso de necesitar una predicción *dura* se quedará con su argumento máximo.

Finalmente hemos analizado la distribución predictiva de un modelo relativamente simple. Incluso en modelos escalares, que propongan **distribuciones conjugadas a priori**, el estudio analítico es complejo. Esto demuestra la necesidad de contar con métodos numéricos que permitan abordar el tema.

6.4. Monte Carlo por Cadenas de Markov (MCMC)

Efectuar el cálculo con distribuciones predictivas, evaluando la integral directamente, puede ser computacionalmente inviable ya que rara vez se cuenta con una forma cerrada o conocida para la distribución a posteriori. Para resolver este problema, se plantea una solución Monte Carlo: combinar métodos de muestreo con la **ley de los grandes números**.

$$\mathbb{E}\left[p(x_{\text{test}}|T)|\mathbf{X} = \mathbf{x}\right] \approx \frac{1}{t_{\max}} \sum_{t=1}^{t_{\max}} p_{X|T=\theta_t}(x_{\text{test}}) \tag{6.63}$$

donde $\theta_1, \cdots, \theta_{t_{\max}}$ son muestras generadas a partir de la distribución *a posteriori*.

En capítulos anteriores se discutió por que la inteligencia artificial no es más que una mezcla de patrones encontrados en datos observados con ruidos aleatorios. Sin embargo, la ley de los grandes números nos muestra el potencial de los datos: cualquier comportamiento esperado puede aproximarse tan bien como uno desee si se cuenta la cantidad de datos suficiente. Este resultado da que pensar acerca de como y cuando entregamos nuestros datos. Quizás estemos alimentando nuestro reemplazo.

El problema con este método es que difícilmente podamos generar muchas muestras independientes. Es entonces cuando surge el Muestreo Monte Carlo por Cadenas de Markov (MCMC) como una alternativa [20, Capítulo 11]. Una **cadena de Markov** es una secuencia de variables aleatorias $\{\theta_t\}_{t\in\mathbb{N}}$ que cumple la propiedad de *falta de memoria*: la distribución del próximo estado depende únicamente del estado actual, y no del pasado completo. Formalmente,

$$p(\theta_{t+1}|\theta_t, \theta_{t-1}\cdots, \theta_0) = p(\theta_{t+1}|\theta_t) = P(\theta_t \to \theta_{t+1}) \tag{6.64}$$

donde $P(\theta_t \to \theta_{t+1})$ es la densidad de transición de θ_t a θ_{t+1} (en la estadística bayesiana todas estas distribuciones serán computadas implícitamente *a posteriori* de observar los datos de entrenamiento). Cuando las probabilidades de transición no dependen del tiempo t, se dice que la cadena es **homogénea**. Una distribución π se dice *estacionaria* para una cadena de Markov si no varía su estadística al propagarse por la cadena;

$$\pi(\theta') = \int \pi(\theta)P(\theta \to \theta')\, d\theta \tag{6.65}$$

es decir, que la distribución de θ y θ' sea la misma (que la distribución de θ no varíe al efectuar un paso en la cadena). Una condición suficiente para asegurar esto es que la conjunta de ambas sea simétrica:

$$\pi(\theta)P(\theta \to \theta') = \pi(\theta')P(\theta' \to \theta) \tag{6.66}$$

la cual se cumple de forma trivial si $\theta = \theta'$ (repetir el estado actual no afecta la condición (6.66) correspondiente al estado estacionario).

La idea general del MCMC es construir una cadena de Markov homogénea $\{\theta_t\}_{t\in\mathbb{N}}$ cuya distribución estacionaria sea la *distribución a posteriori*. Bajo ciertas condiciones, el **teorema de ergodicidad** garantiza que el promedio de los valores generados por la cadena converge a la esperanza, dando por válida (6.63) aunque no se trate de muestras independientes. Para una cadena de Markov homogenea, dichas condiciones se pueden resumir en:

- Irreducible: La transición $\theta \to \theta'$ debe poder ser alcanzada en una cantidad finita de pasos para todo θ y θ'.
- Áperiódica: La transición $\theta \to \theta$ (mantener el estado actual) debe tener probabilidad positiva.
- Recurrente positiva: El tiempo esperado para volver al estado actual es finito.

Los algoritmos utilizados para los modelos bayesianos fueron construidos para cumplir con estas condiciones. A continuación se presentará un ejemplo de cálculo para calcular el estado estacionario de una cadena de Markov homogénea.

Ejemplo 6.2 *Encontrar el estado estacionario de una cadena de Markov homogénea, donde θ es una variable discreta que toma valores en $\{0, 1, 2\}$ y las probabilidades de transición se definen con la matriz* $P = \begin{pmatrix} 0.7 & 0.2 & 0.1 \\ 0.4 & 0.6 & 0.0 \\ 0.0 & 0.9 & 0.1 \end{pmatrix}$.

No es difícil probar que esta cadena es irreducible, aperiódica y recurrente positiva. Toda cadena de Markov irreducible en un espacio de estados finitos tiene una distribución estacionaria única. Se desea encontrar $\pi(0)$, $\pi(1)$ y $\pi(2)$ (representadas por un vector π), tal que se cumpla (6.65). Para variables discretas y finitas eso se puede representar como $\pi = P \cdot \pi$ con $\mathbf{1}^T \cdot \pi = 1$, donde $\mathbf{1}$ es un vector con todas sus entradas en 1. Escribiendo todo eso como un sistema de ecuaciones se obtiene

$$\begin{pmatrix} 0.7 & 0.2 & 0.1 \\ 0.4 & 0.6 & 0 \\ 0 & 0.9 & 0.1 \\ 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \pi(0) \\ \pi(1) \\ \pi(2) \end{pmatrix} = \begin{pmatrix} \pi(0) \\ \pi(1) \\ \pi(2) \\ 1 \end{pmatrix} \tag{6.67}$$

La segunda ecuación $0.4\pi(0) + 0.6\pi(1) = \pi(1)$ implica que $\pi(0) = \pi(1)$. De la tercera $0.9\pi(1) + 0.1\pi(2) = \pi(2)$ se obtiene que $\pi(1) = \pi(2)$. Reemplazando podemos ver que se cumple la primera ecuación $0.7\pi(0) + 0.2\pi(1) + 0.1\pi(2) = \pi(0)$ y, dado que deben sumar 1 (cuarta ecuación), se obtiene $\pi(0) = \pi(1) = \pi(2) = \frac{1}{3}$.

6.4.1. Algoritmos de Muestreo MCMC

El objetivo de los algoritmos de muestreo es generar un proceso cuya secuencia de muestras sea ergódica. Esto no siempre es sencillo, ya que puede no disponerse de toda la información necesaria sobre las distribuciones.

En la Fig. 6.7 puede verse un ejemplo de experimento de muestreo. Se denomina *tune* a la cantidad de muestras a descartar para considerar que se alcanzó el estado estacionario, y *draws* a la cantidad de muestras efectivas que fueron generadas. Se suelen generar varias cadenas y verificar los resultados en cada una (cantidad definida en *chains*). Una disparidad de resultados en las cadenas evidencia que no se alcanzó el mencionado estado estacionario. Dependiendo del tipo de variable aleatoria a muestrear, conviene usar una u otra estrategia. A continuación se presentarán algunos de los muestreos más habituales.

6.4.1.1. Muestreo de Gibbs

Supongamos que, debido a su complejidad, no podemos simular muestras de $\pi(x, y)$, pero que sí es posible generar muestras de las condicionales $\pi(x|y)$ y $\pi(y|x)$ (conocidas y

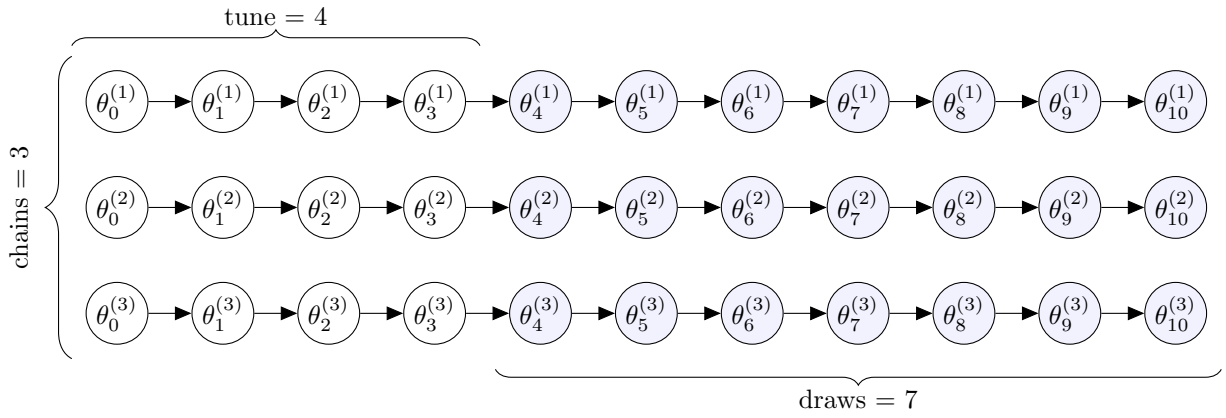


Figura 6.7: Ejemplo de experimento de muestreo. En este caso se simularon tres cadenas independientes (chains = 3), se esperaron cuatro pasos para considerar que se alcanzó el estado estacionario (tune = 4) y se recolectaron siete muestras efectivas en el presunto estado estacionario (draws = 7).

fáciles de muestrear). El muestreo de Gibbs consiste en, a partir de un x_0, iterar alternadamente entre $y_t \sim \pi(y|x_t)$ y $x_{t+1} \sim \pi(x|y_t)$. Luego de suficientes pasos, al alcanzar el estado estacionario, los pares (x, y) estarán distribuidos por $\pi(x, y)$. En la siguiente sección se demostrará por qué este proceso cumple (6.66).

Esta técnica se extiende a más dimensiones muestreando de una componente a la vez. El problema con este muestreo es que es necesario conocer perfectamente todas las distribuciones condicionales, lo cual en la práctica suele ser problemático en muchos casos. Una excepción importante la constituyen las variables Bernoulli, donde se pueden computar todas las probabilidades necesarias.

A continuación se demostrará que el muestreo de Gibbs posee a la distribución a posteriori como estado estacionario y se mostrará un ejemplo de aplicación.

Demostración 6.1 (Estado estacionario) *Notar que, en esta cadena de Markov donde $\theta = (x, y)$, el proceso evoluciona como*

$$(x_0, y_0) \to (x_1, y_0) \to (x_1, y_1) \to (x_2, y_1) \to (x_2, y_2) \to (x_3, y_2) \to (x_3, y_3) \to \cdots \tag{6.68}$$

Para corroborar que cumple la condición suficiente de estacionariedad (6.66) *tomemos un paso de Gibbs $(x_t, y_t) \to (x_{t+1}, y_t)$:*

$$\pi(x_t, y_t)\pi(x_{t+1}|y_t) = \pi(x_{t+1}, y_t)\pi(x_t|y_t) \tag{6.69}$$

donde todas las distribuciones son computadas a posteriori *(la notación queda implícita). Es fácil notar que, si la medida de probabilidad es la misma, la identidad* (6.69) *representa la misma distribución conjunta de (x_t, y_t, x_{t+1}) en ambos lados de la igualdad.*

Ejemplo 6.3 *Sea la distribución correspondiente al modelo gráfico:*

$$P(x,y) \propto e^{x^T a + y^T b + x^T W y} \cdot \mathbf{1}\{x \in \{0,1\}^{d_x}, y \in \{0,1\}^{d_y}\} \tag{6.70}$$

donde $\{a, b, w\}$ son valores conocidos. Explicar el procedimiento para muestrear esta distribución utilizando muestreo de Gibbs.

Este tipo de distribución se conoce como modelo de Boltzmann restringido. Supongamos que buscamos muestrear esta distribución utilizando el muestreo de Gibbs. Se puede ver que la constante de proporcionalidad sería computacionalmente pesada de buscarla para altas dimensiones, pero sus condicionales son más sencillas. Empezando por $P(y|x)$, se puede ver que sus componentes son independientes:

$$P(y|x) \propto e^{(b+W^T x)^T y} \cdot \mathbf{1}\{y \in \{0,1\}^{d_y}\} = \prod_{j=1}^{d_y} e^{[b+W^T x]_j y_j} \cdot \mathbf{1}\{y_j \in \{0,1\}\} \tag{6.71}$$

y por lo tanto $P(y_j = 1|x) \propto e^{[b+W^T x]_j}$ y $P(y_j = 0|x) \propto 1$. Juntando esta información puede verse que $y|x \sim \text{Ber}(\sigma(b + W^T x))$[2] es una vector Bernoulli de componentes independientes. Análogamente, puede deducirse que $x|y \sim \text{Ber}(\sigma(a + Wy))$.

En este caso, podrían generarse muestras inicializando x_0 y $k = 0$ e iterando entre:

- Se genera un y_k a partir de $\text{Ber}(\sigma(b + W^T x_k))$.
- Se genera un x_{k+1} a partir de $\text{Ber}(\sigma(a + W y_k))$.
- $k \leftarrow k + 1$.

6.4.1.2. Muestreo Metropolis

El muestreo Metropolis es usado típicamente en variables aleatorias discretas no binarias (como Poisson, geométrica, hipergeométrica, etc.) que toman valores en los enteros $\mathbb{Z}$, así como también en variables continuas donde no hay diferenciabilidad debido al modelo. Su característica esencial es que solamente le basta con conocer la distribución (conjunta, de todos los parámetros simultáneamente) salvo una constante de proporcionalidad. Es decir que si $\pi(\theta) = \frac{f(\theta)}{Z}$ con $Z > 0$, es suficiente con conocer $f(\theta)$ (que habitualmente se plantea como distribución *a priori* por la verosimilitud).

Este tipo de muestreo propone transicionar de θ_t a θ_{t+1} con el siguiente algoritmo simétrico:

1. Se genera una $\theta' = \theta_t + \delta$, donde δ es una variable aleatoria; en el caso que θ' no esté en el soporte de la variable (por ejemplo una Poisson no puede tomar valores negativos), se repite el proceso. Para el caso discreto, típicamente se propone que

[2] La función sigmoide es $\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{e^z+1}$.

$\delta \sim \mathcal{U}\{-1, 0, 1\}$ (uniforme discreta de 3 átomos) y para el caso continuo se propone $\delta \sim \mathcal{N}(0, \sigma^2)$.

2. Se sortea una variable aleatoria Bernoulli de probabilidad $\alpha(\theta_t, \theta')$. Si dicha variable vale 1, $\theta_{t+1} = \theta'$. Caso contrario $\theta_{t+1} = \theta_t$.

donde

$$\alpha(\theta_a, \theta_b) = \text{mín}\left\{1, \frac{f(\theta_b)}{f(\theta_a)}\right\} \tag{6.72}$$

con $\pi(\theta) = \frac{f(\theta)}{Z}$ (no depende de la constante de normalización). A continuación se efectuará un análisis para demostrará que la distribución *a posteriori* cumple la condición de estacionariedad dada por (6.66).

Demostración 6.2 (Estado Estacionario - Caso Discreto) *La probabilidad de transición del caso discreto descripta anteriormente es de la forma*

$$P(\theta_t \to \theta_{t+1}) = \begin{cases} \frac{\alpha(\theta_t, \theta_t - 1)}{3} & \theta_{t+1} = \theta_t - 1 \\ \frac{\alpha(\theta_t, \theta_t + 1)}{3} & \theta_{t+1} = \theta_t + 1 \\ 1 - \frac{\alpha(\theta_t, \theta_t - 1) + \alpha(\theta_t, \theta_t + 1)}{3} & \theta_{t+1} = \theta_t \\ 0 & \textit{Otros} \end{cases} \tag{6.73}$$

El único caso no trivial que vale la pena analizar en (6.66) *es que ocurre si* $\theta_{t+1} = \theta_t \pm 1$*; en los demás casos, la condición se cumple automáticamente. En este caso*

$$\pi(\theta_t)\frac{\alpha(\theta_t, \theta_{t+1})}{3} = \pi(\theta_{t+1})\frac{\alpha(\theta_{t+1}, \theta_t)}{3} \tag{6.74}$$

Se puede ver que si $\pi(\theta) = \frac{f(\theta)}{Z}$*, la ecuación en cuestión puede reducirse a*

$$f(\theta_t)\alpha(\theta_t, \theta_{t+1}) = f(\theta_{t+1})\alpha(\theta_{t+1}, \theta_t) \tag{6.75}$$

Utilizando el $\alpha(\theta_a, \theta_b)$ *definido en* (6.72)*, se puede comprobar la identidad* (6.75)*. El primer término cumple que* $f(\theta_t)\alpha(\theta_t, \theta_{t+1}) = \text{mín}\{f(\theta_t), f(\theta_{t+1}\}$*, y de forma análoga para el otro término* $f(\theta_{t+1})\alpha(\theta_{t+1}, \theta_t) = \text{mín}\{f(\theta_{t+1}), f(\theta_t)\}$*. De esta manera, queda garantizando que la distribución* a posteriori *es un estado estacionario de la cadena.*

Demostración 6.3 (Estado Estacionario - Caso Continuo) *La transición del caso continuo tiene una distribución mixta: una mezcla entre una normal y una masa puntual (delta de Dirac) en* $\theta_{t+1} = \theta_t$*. Dado que el único caso relevante a chequear de* (6.66) *es el caso donde* $\theta_{t+1} \neq \theta_t$*, basta con analizar la parte continua:* $P(\theta_t \to \theta_{t+1}) = \alpha(\theta_t, \theta_{t+1}) \cdot \mathcal{N}(\theta_{t+1}|\theta_t, \sigma^2)$ *donde* $\mathcal{N}(x|\mu, \sigma^2)$ *hace referencia a la densidad de*

una normal de parámetros μ y σ^2 evaluada en x. En este caso la condición (6.66) *puede escribirse como:*

$$\pi(\theta_t) \cdot \alpha(\theta_t, \theta_{t+1}) \cdot \mathcal{N}(\theta_{t+1}|\theta_t, \sigma^2) = \pi(\theta_{t+1}) \cdot \alpha(\theta_{t+1}, \theta_t) \cdot \mathcal{N}(\theta_t|\theta_{t+1}, \sigma^2) \tag{6.76}$$

La condición anterior puede reducirse a $f(\theta_t)\alpha(\theta_t, \theta_{t+1}) = f(\theta_{t+1})\alpha(\theta_{t+1}, \theta_t)$ usando que la normal es simétrica respecto a la media $\mathcal{N}(\theta_t|\theta_{t+1}, \sigma^2) = \mathcal{N}(\theta_{t+1}|\theta_t, \sigma^2)$. De esta manera llegamos a la misma identidad que en el caso discreto y por lo tanto, podemos concluir que la distribución a posteriori *cumple la condición de estacionariedad* (6.66).

6.4.1.3. NUTS (No-U-Turn Sampler)

El algoritmo NUTS (No-U-Turn Sampler) es un método de muestreo, basado en Metrópolis, para variables aleatorias continuas con distribución a posteriori diferenciable. La versión completa del algoritmo [26] puede verse en la Fig. 6.8; a continuación se presentarán las ideas generales. En lugar de sumar un ruido aleatorio, el algoritmo introduce una variable auxiliar $r \in \mathbb{R}^d$ cuyo objetivo es actuar como dirección de exploración de la distribución *a posteriori*. Esta variable se genera en cada iteración a partir de una distribución normal estándar multivariada. A partir de ella se define la función de energía como:

$$H(\theta, r) = -\log \pi(\theta) + \frac{1}{2}\|r\|^2 \tag{6.77}$$

Esta función de energía combina la log-posterior de θ con una penalización cuadrática sobre r y tiene un papel central en la determinación de la calidad de las propuestas. En cada iteración, partiendo del estado actual (θ_t, r), el algoritmo genera una secuencia de nuevas propuestas (θ, r) mediante un método numérico que utiliza la información del gradiente de la energía. Este procedimiento se conoce como integración tipo **leapfrog**, y consiste en aplicar transformaciones que mantengan aproximadamente constante el valor de $H(\theta, r)$. Así como el gradiente descendente desplaza los parámetros hacia la dirección de decrecimiento de la función objetivo, la integración *leapfrog* desplaza los parámetros sobre una curva de nivel de $H(\theta, r)$. Durante la etapa `tune`, *leapfrog* ajustará el paso del algoritmo para que el nivel de aceptación de un nuevo estado sea `target_accept` (parámetro predefinido por el usuario).

Una característica distintiva de NUTS es que, en lugar de requerir un número fijo de pasos de integración (como en otros algoritmos relacionados), a partir de θ_t, construye dinámicamente un árbol de posibles (θ, r), expandiéndose hacia adelante y hacia atrás, hasta que detecta que continuar expandiendo llevaría a una región ya visitada o a una dirección contraria a la actual, según un criterio geométrico. Esta condición de detención

Algorithm 6 No-U-Turn Sampler with Dual Averaging

Given θ^0, δ, $\mathcal{L}, M, M^{\text{adapt}}$:
Set $\epsilon_0 = \text{FindReasonableEpsilon}(\theta), \mu = \log(10\epsilon_0), \bar{\epsilon}_0 = 1, \bar{H}_0 = 0, \gamma = 0.05, t_0 = 10, \kappa = 0.75$.
for $m = 1$ to M **do**
  Sample $r^0 \sim \mathcal{N}(0, I)$.
  Resample $u \sim \text{Uniform}([0, \exp\{\mathcal{L}(\theta^{m-1} - \frac{1}{2} r^0 \cdot r^0\}])$
  Initialize $\theta^- = \theta^{m-1}$, $\theta^+ = \theta^{m-1}$, $r^- = r^0, r^+ = r^0, j = 0, \theta^m = \theta^{m-1}, n = 1, s = 1$.
  while $s = 1$ **do**
    Choose a direction $v_j \sim \text{Uniform}(\{-1, 1\})$.
    if $v_j = -1$ **then**
      $\theta^-, r^-, -, -, \theta', n', s', \alpha, n_\alpha \leftarrow \text{BuildTree}(\theta^-, r^-, u, v_j, j, \epsilon_{m-1}\theta^{m-1}, r^0)$.
    else
      $-, -, \theta^+, r^+, \theta', n', s', \alpha, n_\alpha \leftarrow \text{BuildTree}(\theta^+, r^+, u, v_j, j, \epsilon_{m-1}, \theta^{m-1}, r^0)$.
    end if
    if $s' = 1$ **then**
      With probability $\min\{1, \frac{n'}{n}\}$, set $\theta^m \leftarrow \theta'$.
    end if
    $n \leftarrow n + n'$.
    $s \leftarrow s' \mathbb{I}[(\theta^+ - \theta^-) \cdot r^- \geq 0] \mathbb{I}[(\theta^+ - \theta^-) \cdot r^+ \geq 0]$.
    $j \leftarrow j + 1$.
  end while
  if $m \leq M^{\text{adapt}}$ **then**
    Set $\bar{H}_m = \left(1 - \frac{1}{m + t_0}\right) \bar{H}_{m-1} + \frac{1}{m+t_0}(\delta - \frac{\alpha}{n_\alpha})$.
    Set $\log \epsilon_m = \mu - \frac{\sqrt{m}}{\gamma} \bar{H}_m, \log \bar{\epsilon}_m = m^{-\kappa} \log \epsilon_m + (1 - m^{-\kappa}) \log \bar{\epsilon}_{m-1}$.
  else
    Set $\epsilon_m = \bar{\epsilon}_{M^{\text{adapt}}}$.
  end if
end for

function $\text{BuildTree}(\theta, r, u, v, j, \epsilon, \theta^0, r^0)$
if $j = 0$ **then**
  Base case—take one leapfrog step in the direction v.
  $\theta', r' \leftarrow \text{Leapfrog}(\theta, r, v\epsilon)$.
  $n' \leftarrow \mathbb{I}[u \leq \exp\{\mathcal{L}(\theta') - \frac{1}{2} r' \cdot r'\}]$.
  $s' \leftarrow \mathbb{I}[u < \exp\{\Delta_{\max} + \mathcal{L}(\theta') - \frac{1}{2} r' \cdot r'\}]$.
  return $\theta', r', \theta', r', \theta', n', s', \min\{1, \exp\{\mathcal{L}(\theta') - \frac{1}{2} r' \cdot r' - \mathcal{L}(\theta^0) + \frac{1}{2} r^0 \cdot r^0\}\}, 1$.
else
  Recursion—implicitly build the left and right subtrees.
  $\theta^-, r^-, \theta^+, r^+, \theta', n', s', \alpha', n'_\alpha \leftarrow \text{BuildTree}(\theta, r, u, v, j - 1, \epsilon, \theta^0, r^0)$.
  if $s' = 1$ **then**
    if $v = -1$ **then**
      $\theta^-, r^-, -, -, \theta'', n'', s'', \alpha'', n''_\alpha \leftarrow \text{BuildTree}(\theta^-, r^-, u, v, j - 1, \epsilon, \theta^0, r^0)$.
    else
      $-, -, \theta^+, r^+, \theta'', n'', s'', \alpha'', n''_\alpha \leftarrow \text{BuildTree}(\theta^+, r^+, u, v, j - 1, \epsilon, \theta^0, r^0)$.
    end if
    With probability $\frac{n''}{n' + n''}$, set $\theta' \leftarrow \theta''$.
    Set $\alpha' \leftarrow \alpha' + \alpha''$, $n'_\alpha \leftarrow n'_\alpha + n''_\alpha$.
    $s' \leftarrow s'' \mathbb{I}[(\theta^+ - \theta^-) \cdot r^- \geq 0] \mathbb{I}[(\theta^+ - \theta^-) \cdot r^+ \geq 0]$
    $n' \leftarrow n' + n''$
  end if
  return $\theta^-, r^-, \theta^+, r^+, \theta', n', s', \alpha', n'_\alpha$.
end if

Figura 6.8: Algoritmo NUTS presentado por Hoffman en [26].

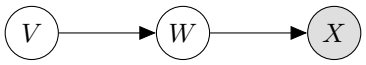


Figura 6.9: Ejemplo de modelado de 3 variables $V \to W \to X$, donde la única variable observable es X.

se conoce como la "No-U-Turn". Una vez elegido un candidato (θ', r'), se acepta el cambio $\theta_{t+1} = \theta'$ con una determinada probabilidad, por el contrario, se decide por $\theta_{t+1} = \theta_t$.

Una vez finalizado este período, el algoritmo congela sus parámetros de control y comienza a generar las muestras válidas. En el contexto de modelos con parámetros continuos, NUTS es especialmente eficiente en espacios de alta dimensión, ya que genera propuestas informadas por la geometría local de la distribución, evitando caminatas aleatorias ineficientes y generando muestras con menor autocorrelación. Esto lo convierte en el algoritmo por defecto en muchas librerías bayesianas modernas como `PyMC`.

6.4.1.4. Ejemplo de Modelo Complejo

En la Fig. 6.9 se muestra un ejemplo de modelado con tres variables $V \to W \to X$, donde la única variable observable es $X = x$. Supongamos que a priori $V \sim \exp(2)$, que $W|_{V=v} \sim \text{Poi}(v)$ y que $X|_{W=w} \sim \chi^2(w+1)$. En este caso, las cadenas se generan con el siguiente procedimiento.

1. Se inicializa v_0, usualmente utilizando una muestra de la distribución a priori exp(2).

2. Se inicializa w_0, a partir de su distribución latente $\text{Poi}(v_0)$ o eventualmente con Poi(0.5) (usando la media de V en lugar de su valor).

3. Se actualiza V, definiendo v_1. Como V es una variable aleatoria continua con densidad derivable, habitualmente se utilizará NUTS aplicando sobre la distribución *a posteriori* $\pi(v, w_0) \propto p(x|w_0)P(w_0|v)p(v) \propto \text{Poi}(w_0|v) \cdot \exp(v|2)$, donde $\text{Poi}(\cdot|\mu)$ es la función de probabilidad de una Poisson de media μ y $\exp(\cdot|\lambda)$ es la función de densidad de una exponencial de intensidad λ. En este caso se absorbió la verosimilitud como constante de proporcionalidad por no depender v.

4. Se actualiza W, definiendo w_1. Al tratarse de una variable discreta, se recomienda utilizar el muestreo Metrópolis. Para el muestreo se utilizará la distribución *a posteriori* $\pi(v_1, w) \propto p(x|w)P(w|v_1)p(v_1) = \chi^2(x|w+1) \cdot \text{Poi}(w|v_1)$ donde $\chi^2(\cdot|\nu)$ es la función de densidad de una *chi-cuadrado* de ν grados de libertad. En este caso se absorbió la distribución a priori $p(v_1)$ como constante de proporcionalidad por no depender w.

5. Se repite el paso (**3**) definiendo v_2 y se continúa iterando la cantidad de pasos que sea necesario.

6.4.2. Calidad de las muestras

Para evaluar la calidad de las muestras de un experimento de MCMC, suelen considerarse dos propiedades: la ergodicidad y la estacionariedad. Gracias al teorema de ergodicidad, podemos aproximar esperanzas a partir de promedios sin pretender que las muestras sean independientes. El problema radica en que la velocidad de convergencia y la varianza de dicho promedio no son las mismas que en el caso de variables independientes. En MCMC, se denomina tamaño de muestra efectiva (Effective Sample Size, ESS) a la cantidad de datos independientes necesarios para alcanzar la misma varianza que posee el promedio de las muestras. Se define como

$$\text{ESS} = \frac{t_{\max}}{1 + 2\sum_{t=1}^{k_{\max}} \rho_t} \tag{6.78}$$

donde ρ_t es la autocorrelación de la cadena y $k_{\max}$ es el valor a partir del cual las autocorrelaciones se vuelven pequeñas o negativas. La ecuación (6.78) será demostrada más adelante. Esta ESS se conoce como **bulk** y se utiliza para cálculos predictivos. Para intervalos de confianza suele usarse otro ESS denominado **tail**.

Otra característica importante, además de la ergodicidad, es verificar si las muestras fueron generadas una vez alcanzado el estado estacionario. Supongamos que se cuenta con una simulación con varias cadenas independientes. Si todas convergieron a la misma distribución, entonces la varianza entre cadenas debería ser similar a la varianza dentro de cada cadena. Se denomina R-hat (o $\hat{R}$, también conocido como diagnóstico Gelman–Rubin) al cociente entre estas varianzas. Si las cadenas aún no convergieron, habrá más variabilidad entre las mismas, y $\hat{R}$ será mayor que 1. En la práctica suele considerarse $\hat{R} > 1.01$ una señal de alerta y un valor $\hat{R} > 1.1$ suele considerarse un problema a resolver.

Demostración 6.4 (Cálculo de ESS) *En cálculo predictivo numérico, la hipótesis de trabajo es aproximar una esperanza con el promedio de las densidades $p_{X|T=\theta_i}(x)$ en lugar de los parámetros pero, en la práctica, se suele analizar la varianza de los parámetros para estandarizar resultados. No sería particularmente complejo trabajar con las verosimilitudes evaluadas en los parámetros, pero no hay grandes diferencias. En última instancia, bajo ciertas condiciones de regularidad, deberían ser comparables ambas ESS.*

Sea $\theta_1, \cdots, \theta_{t_{max}}$ un conjunto de muestras idénticamente distribuidas y sea $\bar{\theta}$ el promedio de las mismas; es simple ver que la varianza si fueran independientes sería de $\frac{\sigma^2}{ESS}$, donde $\sigma^2 = var(\theta_i)$. En el caso de existir un $\rho_t = \frac{cov(\theta_i, \theta_{i+t})}{\sigma^2}$, la varianza se

puede calcular como:

$$var(\bar{\theta}) = var\left(\frac{1}{t_{max}}\sum_{t=1}^{t_{max}}\theta_t\right) = \frac{1}{t_{max}^2}\sum_{i=1}^{t_{max}}\sum_{j=1}^{t_{max}} cov\,(\theta_i, \theta_j) \tag{6.79}$$

$$= \frac{1}{t_{max}^2}\left(\sigma^2 \cdot t_{max} + 2\sigma^2 \sum_{i=1}^{t_{max}-1}\sum_{j=i+1}^{t_{max}} \rho_{j-i}\right) \tag{6.80}$$

$$= \frac{\sigma^2}{t_{max}}\left(1 + 2\sum_{t=1}^{t_{max}-1}\left(1 - \frac{t}{t_{max}}\right)\cdot \rho_t\right) \tag{6.81}$$

donde se aplicó el cambio de variables $t = j - i$. Bajo las hipótesis usuales en modelos MCMC, es razonable que tanto ρ_t como $1 - \frac{t}{t_{max}}$ se vayan achicando a medida que se avanza en la cadena. Los algoritmos suelen truncar la suma, en un k_{max}, cuando las autocorrelaciones se vuelven pequeñas o negativas. Con este procedimiento suele ocurrir que $k_{max} \ll t_{max}$, por lo que despreciando $\frac{t}{t_{max}}$ se puede aproximar:

$$var(\bar{\theta}) \approx \frac{\sigma^2}{t_{max}}\left(1 + 2\sum_{t=1}^{k_{max}} \rho_t\right) \tag{6.82}$$

Igualando este resultado con la expresión de la varianza para variables independientes $\frac{\sigma^2}{ESS}$, y despejando, se obtiene (6.78) *finalizando la demostración.*

En resumen, cuando se desarrolla un algoritmo MCMC lo primero a analizar es si se alcanzó el estado estacionario observando $\hat{R}$. Caso contrario se aumentará la etapa `tune` o se reparametrizará el modelo (en el caso del muestreo NUTS también se puede modificar el `target_accept`). Una vez alcanzado la performance deseada, se procede a evaluar si se cuenta con suficientes muestras para utilizar el *teorema ergódico* observando el ESS. En caso no ser suficiente, una primera opción sería aumentar las muuestras `draws`. Si el problema persiste, se debe volver al inicio mejorando el estado estacionario.

6.4.3. Introducción a `PyMC`

`PyMC` es una biblioteca de Python para inferencia estadística bayesiana, que permite construir modelos probabilísticos complejos y realizar inferencia sobre ellos de forma automatizada [6]. Es muy intuitivo de usar, el programador debe simplemente describir el grafo a implementar.

Supongamos que queremos implementar el modelo descripto en la Sec. 6.1.2. El mismo puede ser definido mediante el siguiente código

```
import pymc as pm
import numpy as np
```

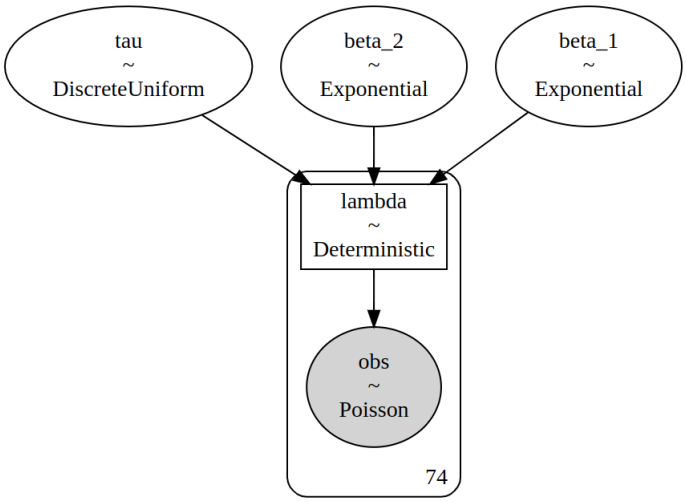


Figura 6.10: Modelo de `PyMC` generado por `model_to_graphviz`.

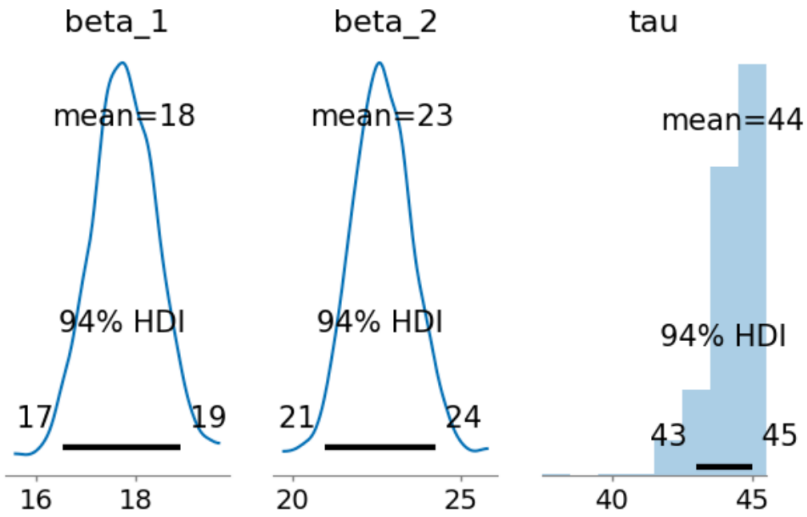


Figura 6.11: Gráfico de las densidades/probabilidades *a posteriori* generado la función `plot_posterior` de `PyMC`.

```
count_data = np.loadtxt("txtdata.csv")
n_count_data = len(count_data)
idx = np.arange(n_count_data) # Index
alpha = 1.0/count_data.mean()
with pm.Model() as model:
  beta_1 = pm.Exponential("beta_1", alpha)
  beta_2 = pm.Exponential("beta_2", alpha)
  tau = pm.DiscreteUniform("tau", lower=0, upper=n_count_data - 1)
  lambda_func = pm.math.switch(tau > idx, beta_1, beta_2)
  lambda_ = pm.Deterministic("lambda",lambda_func)
  observation = pm.Poisson("obs", lambda_, observed=count_data)
```

```
pm.model_to_graphviz(model)
```

donde la última instrucción `pm.model_to_graphviz(model)` graficó la Fig. 6.10 como resultado. El código describe el grafo dentro del *entorno* `Model`, mencionando que datos corresponden a la variable observable (`count_data`). Para muestrear la distribución a posteriori, basta con utilizar el comando `sample` indicando en número de *draws*, *tune* y *chains* deseado.

```
with model:
  trace = pm.sample(draws=1000, tune=1000, chains=2)


import arviz as az
summary = az.summary(trace, var_names=["beta_1","beta_2","tau"])
print(summary)
```

donde la función `summary` nos permite conocer el ESS y el $\hat{R}$ por variable. El objeto `trace` contiene las muestras generadas durante el proceso.

```
beta_1_samples = trace.posterior['beta_1'].values
beta_2_samples = trace.posterior['beta_2'].values
tau_samples = trace.posterior['tau'].values
lambda_samples = trace.posterior['lambda'].values
_ = pm.plot_posterior(trace.posterior[['beta_1','beta_2','tau']])
```

donde la función `plot_posterior` generó el gráfico de las densidades/probabilidades *a posteriori* de la Fig. 6.11.

En el caso de quererse calcular la distribución predictiva o sus derivados (alguna probabilidad, la esperanza, la varianza), se deberán combinar las muestras de `trace` para lograrlo (usualmente se recomienda trabajar cada cadena por separado para corroborar que los resultados sean estacionarios). Pero, puede existir el caso donde lo que efectivamente se desee son muestras de la distribución predictiva[3]. Para ello basta con programar:

```
with model:
  posterior_pred = pm.sample_posterior_predictive(trace,predictions=True)
pred_samples = posterior_pred.predictions['obs'].values
```

La función `sample_posterior_predictive` genera una cantidad de muestras igual a las observadas durante el entrenamiento en cada paso de la cadena. Por lo que si uno desea un solo ejemplo de muestra, podría quedarse con `pred_samples[0,-1]`: De la cadena (0) quedarse con el último (-1) eslabón.

[3]Un muestreo de una distribución aproximada por muestreo. La calidad de estas muestras es menor debido al doble muestreo, por lo que no es recomendable usarlas para estimar la distribución predictiva.

Bibliografía

[1] W. Feller, *An Introduction to Probability Theory and Its Applications, Volume I.* USA: Society for Industrial and Applied Mathematics, 1969.

[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* Wiley Series in Telecommunications and Signal Processing, Wiley-Interscience, 2006.

[3] R. Duda, P. Hart, and D. Stork, *Pattern Classification.* John Wiley, 2 ed., 2001.

[4] G. Casella and R. Berger, *Statistical Inference.* Duxbury advanced series in statistics and decision sciences, Thomson Learning, 2nd ed., 2002.

[5] P. W. Zehna, "Invariance of Maximum Likelihood Estimators," *The Annals of Mathematical Statistics*, vol. 37, no. 3, p. 744, 1966.

[6] C. Davidson-Pilon, *Bayesian Methods for Hackers: Probabilistic Programming and Bayesian Inference.* Addison-Wesley Professional, 1st ed., 2015.

[7] K. P. Murphy, *Machine Learning: A Probabilistic Perspective.* The MIT Press, 2012.

[8] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning.* Springer Series in Statistics, Springer New York Inc., 2001.

[9] K. Petersen and M. Pedersen, *The Matrix Cookbook.* Technical University of Denmark, 2012.

[10] A. Cauchy, "Méthode générale pour la résolution des systèmes d'équations simultanées," *Comp. Rend. Sci. Paris*, vol. 25, pp. 536–538, 1847.

[11] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, April 1997.

[12] T. M. Apostol, *Mathematical analysis.* Addison-Wesley series in mathematics, Reading, MA: Addison-Wesley, 1974.

[13] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming.* New York: Springer, 3rd ed., 2008.

[14] R. A. Horn and C. R. Johnson, *Matrix Analysis.* Cambridge University Press, 2nd ed., 2012.

[15] A. H. Sayed, *Adaptive Filters.* Hoboken, NJ: Wiley–IEEE Press, 1st ed., Apr. 2008. Hardcover.

[16] M. G. González, M. Vera, A. Dreszman, and L. J. Rey Vega, "Diffusion assisted image reconstruction in optoacoustic tomography," *Optics and Lasers in Engineering*, vol. 178, 2024.

[17] C. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[18] L. Ferrer and D. Ramos, "Evaluating posterior probabilities: Decision theory, proper scoring rules, and calibration," *Transactions on Machine Learning Research*, 2025.

[19] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York: Cambridge University Press, 2004.

[20] C. M. Bishop, *Pattern Recognition and Machine Learning*. Information Science and Statistics, Springer-Verlag New York, Inc., 2006.

[21] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[22] J. Peters, D. Janzing, and B. Scholkopf, *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press, 2017.

[23] M. Aigner and G. M. Ziegler, *Proofs from THE BOOK*. Springer Publishing Company, Incorporated, 4th ed., 2009.

[24] M. Vera, L. Rey Vega, and P. Piantanida, "Information flow in Deep Restricted Boltzmann Machines: An analysis of mutual information between inputs and outputs," *Neurocomputing*, vol. 507, pp. 235–246, 2022.

[25] J. von Kügelgen, L. Gresele, and B. Schölkopf, "Simpson's paradox in covid-19 case fatality rates: A mediation analysis of age-related causal effects," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 1, pp. 18–27, 2021.

[26] M. Hoffman and A. Gelman, "The No-U-Turn sampler: adaptively setting path lengths in hamiltonian monte carlo," *Journal of Machine Learning Research*, vol. 15, p. 1593–1623, Jan. 2014.